T.C.
MİLLÎ EĞİTİM BAKANLIĞI

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

PROGRAMLAMA MANTIĞI

Ankara, 2014

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR iii
GİRİŞ 1
ÖĞRENME FAALİYETİ-1 3
1. ALGORİTMA 3
1.1. Bilgisayarda Problem Çözme Adımları 3
1.1.1. Problemi Tanıma 3
1.1.2. Algoritma Geliştirme 4
1.1.3. Girdi Ve Çıktı Biçimi Belirleme 4
1.1.4. Akış Diyagramı Çizme 4
1.1.5. Kodlama 5
1.1.6. Programı Sınama 5
1.2. Algoritma Örnekleri 7
1.3. Algoritmada Sayısal Örnekler 11
1.3.1 Değişkenler 11
1.3.2. Programlamada İşlemler 12
1.4. Programlamada Karar İfadesi 15
1.4.1. Eğer İfadesinin Tek Durumlu Kullanımı 15
1.4.2. Eğer İfadesinin İki Durumlu Kullanımı 16
1.4.3. İç İçe Karar Yapısının Kullanımı 17
1.5. Sayaç Mantığı 20
1.6. Döngüler 22
1.6.1. Temel Döngü Yapısı 22
1.6.2. İç İçe Döngüler 24
UYGULAMA FAALİYETİ 27
ÖLÇME VE DEĞERLENDİRME 29
ÖĞRENME FAALİYETİ-2 30
2. BİLGİSAYAR PROGRAMLAMA 30
2.1. Yazılım Kavramı 30
2.1.1. Bilimsel Ve Mühendislik Yazılımları 30
2.1.2. Mesleki Yazılımlar 31
2.1.3. Yapay Zeka Yazılımları 31
2.1.4. Görüntüsel Yazılımlar 31
2.1.5. Sistem Yazılımları 31
2.2. Sayı Sistemleri 31
2.2.1. İki Tabanlı (Binary) Sayı Sistemi 31
2.2.2. Basamak Değerleri 33
2.2.3. Onaltı Tabanlı (Heksadesimal) Sayı Sistemi 35
2.3. Programlama Dilleri 35
2.4. C Programlama Dili 37
2.4.1. C Programlama Dili Özellikleri 37
2.4.2. C Programının Derlenmesi 38
2.4.3. Program editörünün kullanılması 40
UYGULAMA FAALİYETİ 48
ÖLÇME VE DEĞERLENDİRME 50
MODÜL DEĞERLENDİRME 52

CEVAP ANAHTARLARI 53
KAYNAKÇA 54

AÇIKLAMALAR

ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Ortak Alan**
MODÜLÜN ADI	**Programlama Mantığı**
MODÜLÜN TANIMI	Programlama yapabilme bilgi ve becerilerin verildiği öğretim malzemesidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Bu modülü tamamladığınızda programlamaya hazırlık işlemini yapabileceksiniz.
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında programlamaya hazırlık işlemini doğru ve hatasız olarak yapabileceksiniz. **Amaçlar** **1.** Algoritma hazırlama kurallarına göre akış diyagramını doğru bir biçimde çizebileceksiniz. **2.** Bilgisayar programı yazım aracını hatasız bir şekilde kuracak ve programı derleyebileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Programlama Laboratuarı **Donanım:** Bilgisayar, Bilgisayar çevre birimleri, programlama akış şablonu
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen, modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili öğrenci,

Bugün bilgisayarlarda kullanılan işletim sistemleri, çizim programları veya oyunlar çeşitli program kodlarının bir araya getirilmesi ile oluşmuştur. Siz de temel programlama 1 modülü ile bilgisayar programcılığına adım atmış olacaksınız. Bilgisayar programlamada hangi programlama dilini seçerseniz seçin yapacağınız işlemleri belirli bir mantık üzerine kurmak zorundasınız. O zaman işe programlama mantığını öğrenmeyle başlamak bu yolda atılabilecek en doğru adımdır. Peki bu mantığı nasıl öğreniriz ve en önemlisi de bunu bilgisayara nasıl anlatırız?

Bir bilgisayar ile iletişime geçebilmek için herhangi bir programlama dili kullanmak zorundayız. Hangi programlama dilini kullanırsak kullanalım bilgisayara yaptıracağımız işleri önceden kafamızda tasarlamamız gerekir. Herhangi bir işe başlamadan kafamızda yada bir kağıt üzerinde tasarladığımız işlem yoluma algoritma denir.

Programlamanın en önemli kısmı algoritma hazırlayabilmektir. Algoritma hazırlandıktan sonra hazırlanan algoritmanın herhangi bir programlama dilinde kodlanması işin en basit kısmıdır. Burada önemli olan programlama dili değil, problemin çözümü için algoritma geliştirebilmektir.

Bu nedenle programlamaya hazırlık modülünün ilk öğrenme faaliyetinde algoritma oluşturmayı öğreneceksiniz. İkinci öğrenme faaliyetinde ise bilgisayarda ilk programınızı yazacak ve program çıktısını ekran üzerine göreceksiniz.

Bilgisayarı kendi amacınız için programlamak ve programınızın sonucunu görmek çalışmalarınızda motivasyon sağlayacaktır.

ÖĞRENME FAALİYETİ-1

AMAÇ

Algoritma hazırlama kurallarına göre akış diyagramını doğru bir biçimde çizebileceksiniz.

ARAŞTIRMA

- Günlük yaşamınızda karşılaştığınız problemlere karşı davranışlarınızı düşününüz.
- Problem karşısında uyguladığınız çözüm yollarının planlı olup olmadığını arkadaşlarınız ile paylaşınız.
- Problemi çözmek için takip ettiğiniz adımları sıralayınız.

1. ALGORİTMA

Günlük hayatımızda karşılaştığımız problemlerin çözümünde sağlıklı bir sonuca ulaşabilmenin yolu planlı bir yol izlenme ile mümkün olacaktır. Rastgele çözümler ile süreklilik sağlanamaz. Bilgisayarla problem çözümünde de bu geçerlidir. Problemin analizi, çözümün tasarlanması, gerçekleştirilmesi ve kontrolü çözümün temel aşamalarıdır.

1.1. Bilgisayarda Problem Çözme Adımları

Çözüm için her şeyden önce problemin tam olarak anlaşılması gerekir. Tam anlamıyla tanımlanamayan bir problemin çözümü de yanlış olacak ve istenilen sonuç elde edilemeyecektir.

1.1.1. Problemi Tanıma

Doğru sonucun bulunabilmesi için neden, nasıl, ne için sorularının yanıtlanması gerekir.

Descartes “Discourse on method” isimli kitabında problem çözme ile ilgili şu önerilerde bulunur:

- Doğruluğu kesin kanıtlanmadıkça, hiçbir şeyi doğru olarak kabul etmeyin, tahmin ve önyargılardan kaçının.
- Karşılaştığınız her güçlüğü mümkün olduğunca çok parçaya bölün.
- Düzenli bir biçimde düşünün; anlaşılması en kolay olan şeylerle başlayıp yavaş yavaş daha zor ve karmaşık olanlara doğru ilerleyiniz.

- Olaya bakışınız çok genel, hazırladığınız ayrıntılı liste ise hiçbir şeyi dışarıda bırakmayacak kadar kusursuz ve eksiksiz olsun.

1.1.2. Algoritma Geliştirme

Algoritma, bir sorunun çözümü için izlenecek yoldur. Başka bir ifadeyle mevcut bilgilerden istenilenlere erişme yöntemidir. Genellikle bir sorunun birden fazla çözüm yolu olabilir. Bunlardan en uygunu seçilmeye çalışılır. Her algoritma aşağıdaki kriterleri sağlamalıdır:

- **Giriş :**Her algoritmanın bir başlangıç noktası vardır. Başlangıçta herhangi bir sayısal değer dışarıdan verilmelidir.
- **Çıkış :**Her algoritmanın bir çıkış noktası vardır ve en az bir değer üretmelidir.
- **Kesinlik :**Algoritmanın her adımında yapılacak iş açık olmalı ve farklı anlamlar içermemelidir.
- **Sonluluk :**Her türlü olasılık için algoritma sonlu adımda bitmelidir.

Günlük yaşamımızda her zaman yaptığımız işlemleri algoritmik sırayla aşağıdaki gibi yazabiliriz:

Yatmadan önce saatini ayarla
Alarm çalınca kalk
Elini yüzünü yıka
Kahvaltı hazırla
Kahvaltı yap
Okul kıyafetini giy
Kapıyı aç ve dışarı çık
Biletini al
Otobüse bin
İnmek için ikaz lambasına bas
Okula doğru git
Okula gir

1.1.3. Girdi Ve Çıktı Biçimi Belirleme

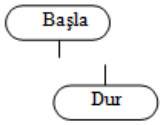
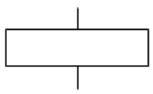
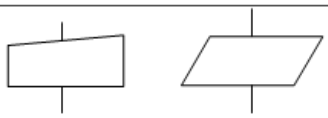
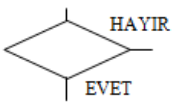
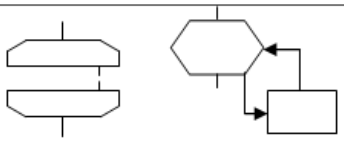
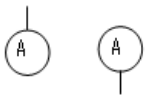
Sonuçların dış ortama, dolayısıyla insana aktarımı düzgün bir biçimde yapılmalıdır. Programcı program çıktısı olarak almak istediği sonucun biçimini tasarlar. Bir sonuç biçimi tasarlanırken anlaşılır ve kullanılabilir olmasına özen gösterilmelidir. Genellikle programın çözdüğü soruna ilişkin bazı verilerin dışarıdan verilmesi gerekir. Örneğin bir denklem takımının kökleri bulunacaksa, ilgili katsayıların programa verilmesi gibi.

1.1.4. Akış Diyagramı Çizme

Akış şeması belirli bir işin yapılabilmesi için, basit işlemlerle şema halinde gösterilmesidir. Kısaca algoritmanın şemalarla gösterilmesidir. Algoritma geliştirildikten sonra, daha iyi anlaşılabilir ve programlama dillerine aktarımının daha kolay olması

nedeniyle, akış şeması haline getirilir. Böylece sorunun çözüm basamakları, birbirleri ile ilişkileri ve bilgi akışı daha kolay görülebilir ve yanlışlıklar düzeltilebilir.

Algoritmaya göre akış diyagramının çizilmesi ve programın akış diyagramına göre yazılması bilgisayar programcılığına yeni başlayanlar için çok önemlidir.

İSİM	SEMBOL	AÇIKLAMA
Başlatma ve Durdurma		Başla akış diyagramının en üst kısmına konmalıdır. Dur ise en altta olmalıdır
Süreç		Bu herhangi bir süreç, fonksiyon veya işlemi ifade eder.
Gösterim		Ekranda görüntüleme
Giriş		Klavyeden veri girişi
Karar verme		Karar verilme noktasındaki bağlantı noktası.
Döngü		Bir komut veya komutlar grubunun belirlenmiş şarta göre tekrarlanması için kullanılır.
Bağlantı		Bir akış diyagramından aynı akış diyagramının başka bir bölümüne geçiş yapmak için kullanılır

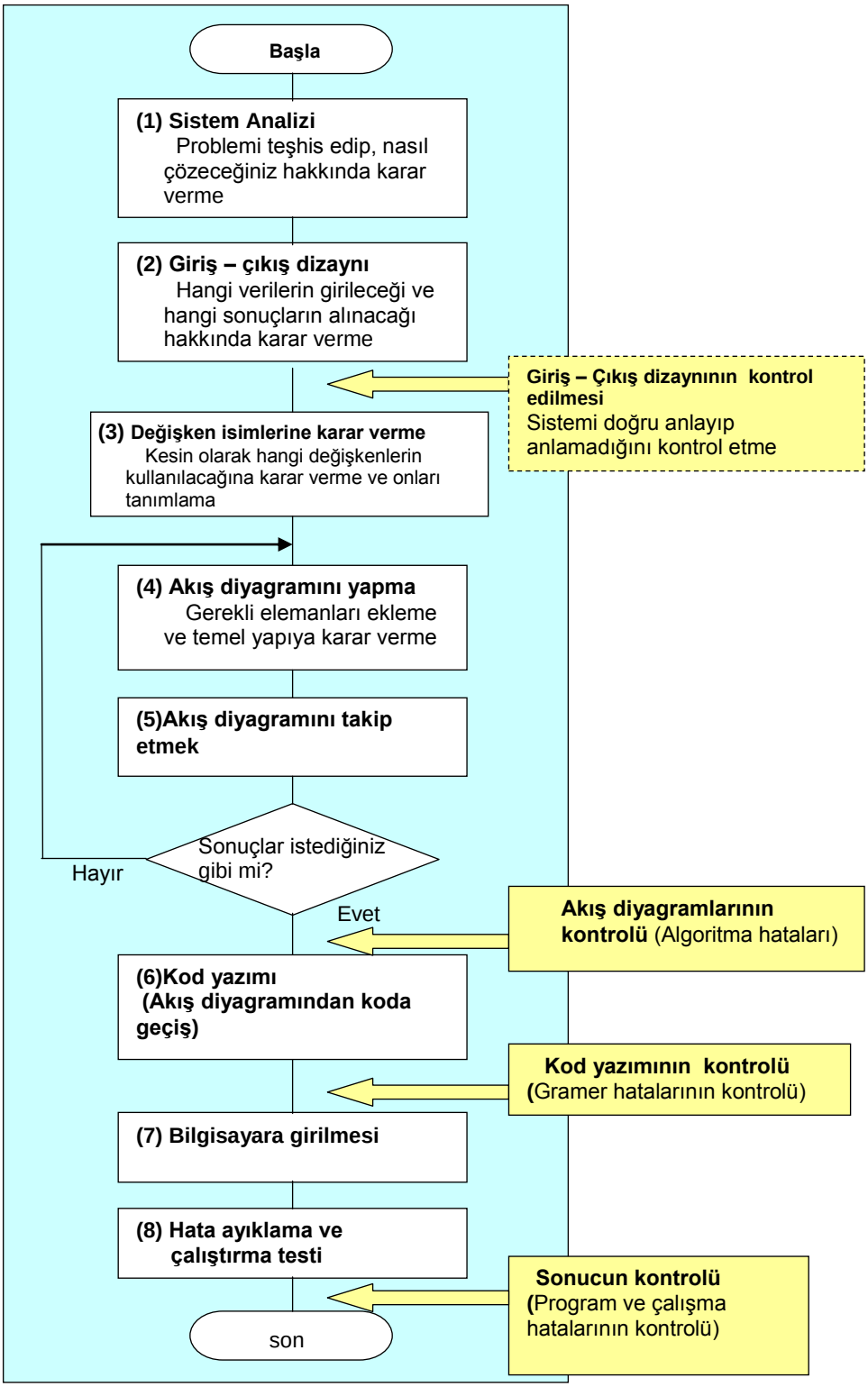
Şekil 1.1: Akış diyagramı sembolleri

1.1.5. Kodlama

Akış şemaları çizildikten sonra, sorunun yapısına uygun bir programlama dili seçilir. Bu dil ile akış şemaları dilin kurallarına uygun olarak bilgisayarın anlayabileceği duruma getirilir.

1.1.6. Programı Sınama

Program yazıldıktan sonra, sonuçları daha önceden bilinen veriler girilerek, eldeki sonuçlarla çıkan sonuçlar karşılaştırılır. Programın doğru çalışıp çalışmadığı sınanır.

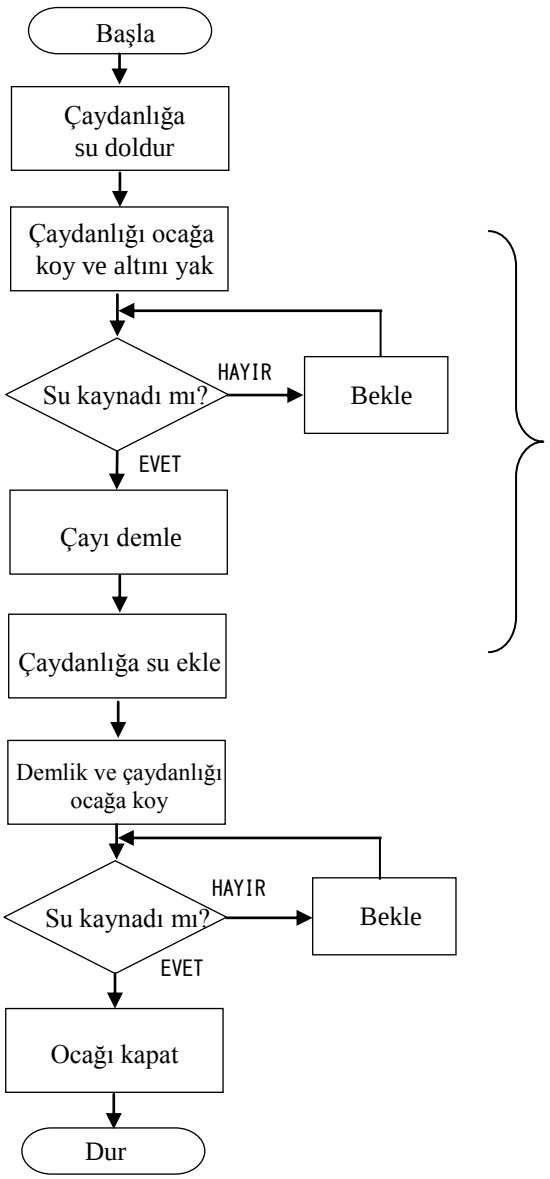


Şekil 1.2: Problem çözme algoritması

1.2. Algoritma Örnekleri

Bu bölümde günlük yaşamınızda her zaman yaptığınız fakat üzerine düşünmediğiniz işlemleri algoritma mantığı ile uygulayacaksınız. Böylece algoritmada sayısal işlemleri daha kolay yapabileceksiniz.

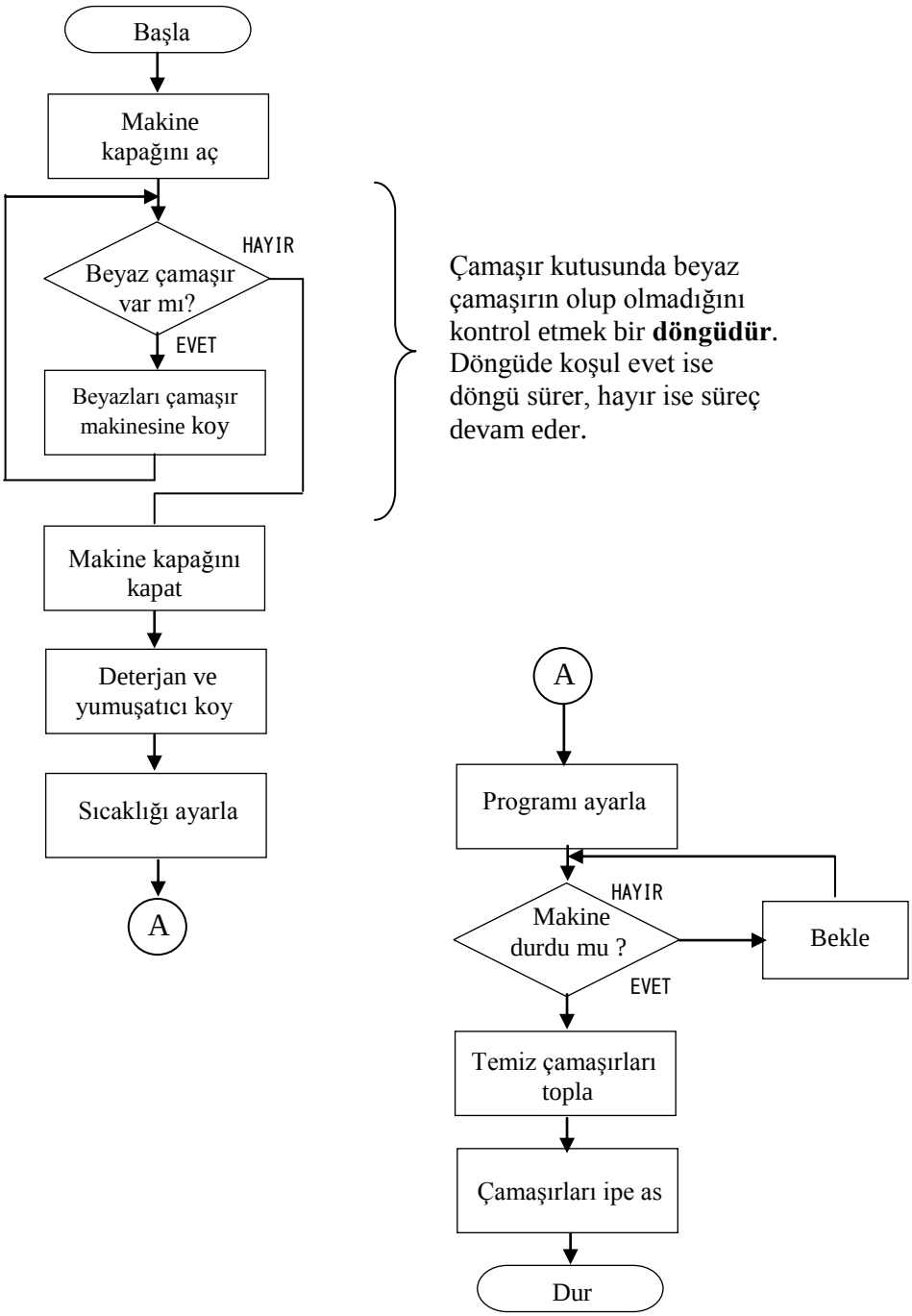
Örnek 1.1.



Çay demleme işlemine başlarken "çaydanlığı ocağa koymak ve altını yakmak" bir süreçtir. Suyun kaynamasını kontrol etmek ise karardır. Karar ifadesi sağlanıyorsa **(EVET)** akış sürer, bu durum gerçekleşmez ise **(HAYIR)** süreç devam eder.

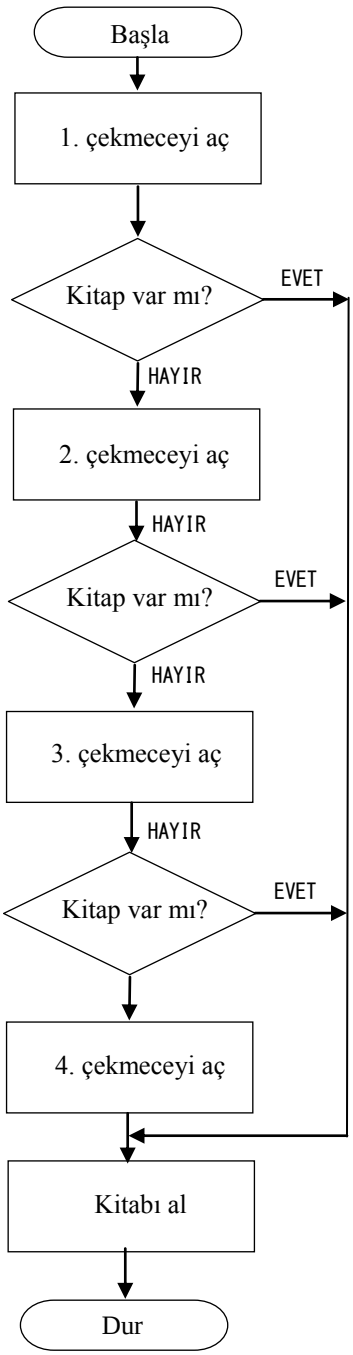
Şekil 1.3: Örnek (Çay demleme) akış diyagramı

Örnek 1.2.



Şekil 1.4: Örnek (Çamaşır yıkama) akış diyagramı

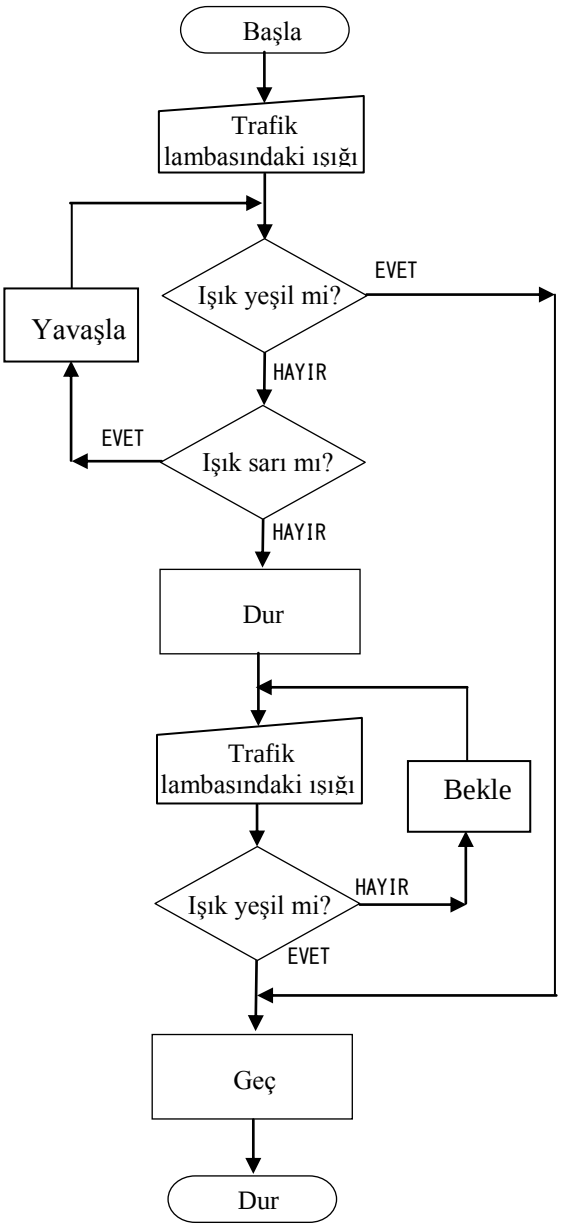
Örnek 1.3.



Kitap arama işleminde çekmeceleri açma sırası değişebilir. Bu örnek için kullanılan karar ifadesinden çıkan sonuç "HAYIR" ise süreç devam eder.Karar "EVET" olduğunda akış sona erecektir.

Şekil 1.5: Örnek (Kitap arama) akış diyagramı

Örnek 1.4.



Araba ile yolculuk yaparken trafik lambaları karşısında davranışlarınız şekildeki gibi olacaktır.
Trafik lambasındaki ışığın yeşil – sarı – kırmızı yanma sırasını dikkate alırsak
Yeşil yanma durumunda akış diyagramında süreç "GEÇ" olacaktır.
Eğer ışık yeşil değil ise sarıdır. Bu durumda yavaşlamamız gerekir.
Işık sarı değil ise kırmızıdır ve durmamız gerekir.
Durduktan sonra okuma işlemi tekrar yapılır ve ışığın kırmızıdan yeşile dönmesi sınanır.

Şekil 1.6: Trafik lambası akış diyagramı

1.3. Algoritmada Sayısal Örnekler

Bu bölümde bilgisayar programlamanın temeli olan algoritmayı sayısal örnekler ile öğreneceksiniz. Verilen örneklerde herhangi bir programlama dili komutu kullanılmayacak, bunun yerine algoritma basit ve yalın ifadeler ile anlatılacaktır.

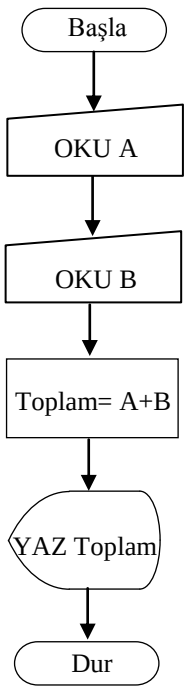
1.3.1 Değişkenler

Programın herhangi bir basamağında tanımlanan ve farklı zamanlarda farklı değerler alabilen sembolik ifadeye değişken denir. Değişkenler bilgisayarın RAM adı verilen belleğinde geçici olarak saklanırlar. Değişkenleri RAM bellekte tahsis edilmiş odacıklar olarak düşünebiliriz. Yani bir değişken tanımlandığında RAM bellekte bir odacık (bölüm) açılır ve buraya değişken ismi ile ulaşılır. Gerektiğinde bu değişkenler içine yeni değerler yazılabilir. Yeni bir değer yazıldığında eski değerler silinir. Programdaki değişkenler problemin tanımı ve girdi – çıktı belirleme aşamalarında belirlenmelidir.

Program yazımı sırasında değişken isimleri verilirken anlamlı kısaltmalar yapmak programın anlaşılabilir olması açısından önemlidir. Uzun değişken adları bellekte daha fazla yer kaplayacağı için programın hacmini arttırır, çok kısa değişkenler de programın izlenmesinde sıkıntılar yaratır. Örneğin;

ortalama → ort **öğrenci notu → ogrnot** **satış fiyatı → sfiyat**

Örnek: Klavyeden girilen iki sayının toplamını bulan algoritmayı tasarlayın.



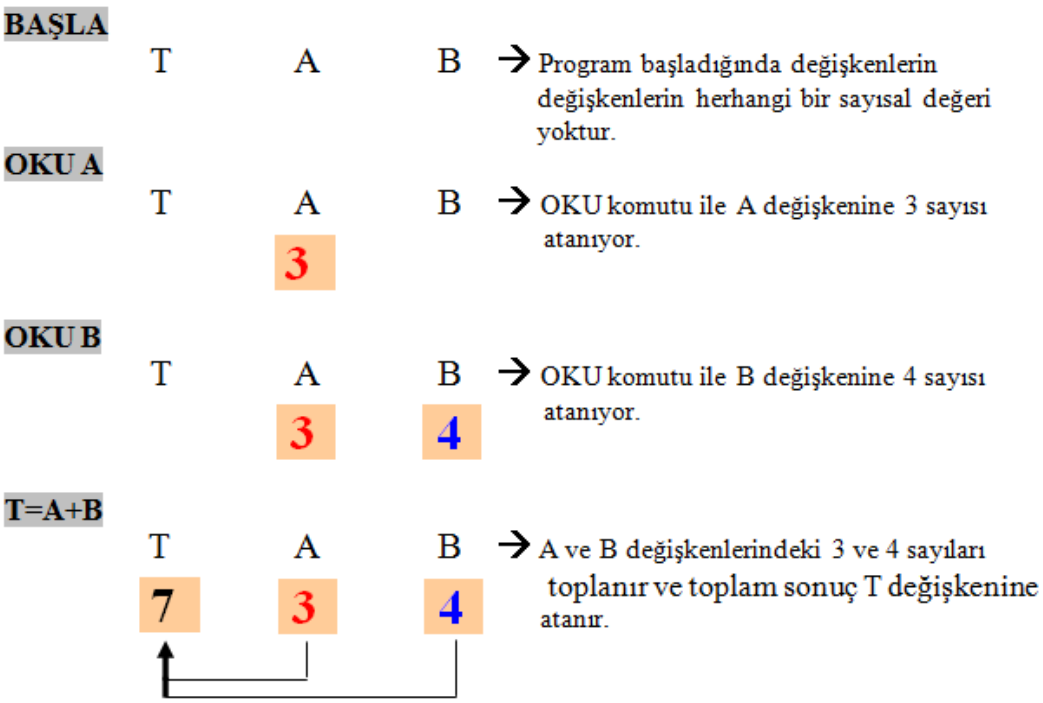
Klavyeden girilen iki sayının toplamını bulan program yazılırken 3 adet değişken tanımlanmalıdır. Çünkü klavyeden 2 sayı girilecek ve bu sayılar toplanarak 3. bir değişkene aktarılacaktır.

Algoritma sadece akış diyagramı ile değil komutlar ile de açıklanabilir. Buna göre akış diyagramının algoritması aşağıdaki şekilde yazılabilir:

BAŞLA
OKU A
OKU B
Toplam=A+B
YAZ Toplam
DUR

Şekil 1.7: İki sayının toplanması

Örnekteki algoritmaya göre değişkenlerin durumu şu şekildedir:

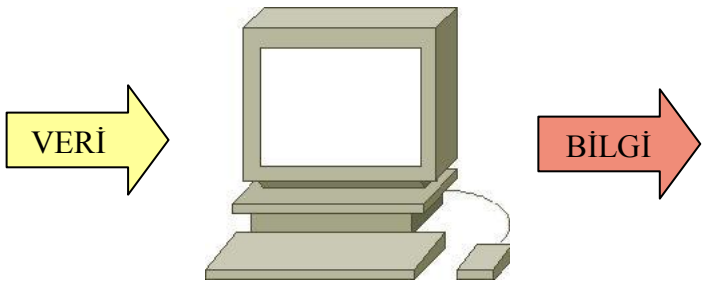


1.3.2. Programlamada İşlemler

Bilgisayarlar bilgileri işleyen makinelerdir. Verilen bilgileri, kabul ederek tarif edilen problemi çözer veya bilgileri kullanarak istenen sonuçları çıkartır. Bilgisayar bilgileri okur, işler, saklar, geri verir. Ancak bütün bu işleri çok hızlı bir şekilde ve güvenilirlikle yapar.

Bu açıklamalara göre bazı kavramları bilmemiz gerekir.

Veri, bilgisayara girilen sayısal değerler veya metinlerdir. Kendi başına anlamı olmayan bu değerler işlenerek gerekli sonuçlar çıkartılır. Bilgi ise verilerin işlenmesi ile çıkartılan sonuçlardır. İnsanlar için anlamlı olan bu sayısal sonuçlar değerlendirilir ve uygulanır.



Şekil 1.8: Veri ve bilgi

Bilgisayarlar girilen veriler ile şu işlemleri yaparlar:

1.3.2.1. Aritmetik Ve Karşılaştırma İşlemleri

Bilgisayarlar temel matematik işlemleri yapabildiği gibi karşılaştırma işlemleri de yapabilirler. Yaşları girilen iki kişiden genç olanı bulmak, sınıfın içinde en yüksek puanı alan öğrenciyi tespit etmek, bir alışveriş merkezinde belli bir tarihte yapılan işlemleri bulmak gibi örnekleri verebiliriz.

Toplama	Çıkarma	Çarpma	Bölme	Aktarma
+	-	*	/	=
Büyük	Küçük	Büyük eşit	Küçük eşit	Eşit değil
>	<	>=	<=	<>

Şekil 1.9: Aritmetik ve karşılaştırma işlem operatörleri

Aritmetik işlemlerde parantezin önceliği vardır.
İşlemde parantez yoksa çarpma ve bölmenin toplama ve çıkarmaya karşı önceliği vardır.

Örnek : $\frac{CD}{AD} + B + \frac{CD}{A} \longrightarrow$ (C*D/(A*D))+B+C*D/A

1.3.2.2.Mantıksal Bağlaçlar

Bilgisayarlar iki veya daha fazla koşulun birlikte sınanmasını da yaparlar.Örneğin puanı "69 dan büyük **VE** 85 den küçük" olma koşulu öğrenci notunun 4 olmasını sağlar.Diğer bir örnek de "öğrencinin edebiyat **VEYA** matematik notunun 70 ten yukarı olması durumunda öğrenci sınıfı geçebilsin" ifadesidir.

VE bağlacı: Verilen koşulların hepsi doğru ise sonuç sağlanır. VE bağlacı ile bağlanmış önermelerden en az birinin yanlış olması sonucu yanlış yapar.

VEYA bağlacı: VEYA bağlacı ile bağlanan koşullardan en az birisi doğru ise sonuç doğrudur.

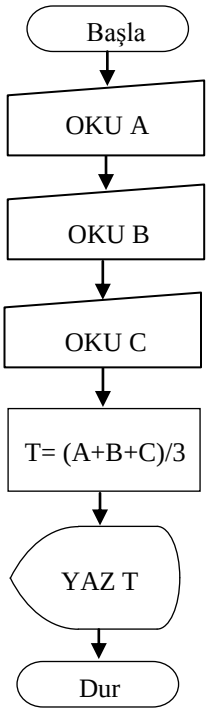
DEĞİL bağlacı: DEĞİL bağlacı doğruyu yanlış, yanlışı doğru yapar. DEĞİL tek bir önerme ve koşul üzerinde uygulanır.

Doğru= 1 ve Yanlış= 0 tanımıyla, x ve y'nin alacağı değerlerin sonuçları aşağıdaki tabloda gösterilmiştir:

X	Y	X **ve** Y	X **veya** Y	Z	**değil** Z
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1		
1	1	1	1		

Şekil 1.10: Mantıksal bağlaçlar

Örnek: Klavyeden girilen 3 sayının ortalamasını bulan algoritmayı tasarlayın.

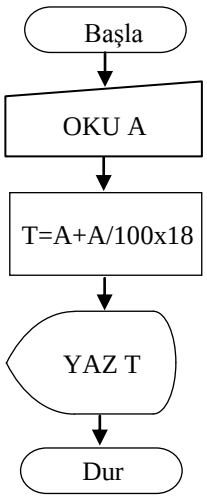


BAŞLA
OKU A
OKU B
OKU C
T=(A+B+C)/3
YAZ T
DUR

Bu örnekteki (A+B+C)/3 işleminde parantez önceliği vardır. Eğer parantez kullanılmasaydı A+B+C/3 işleminde bölmenin toplamaya göre önceliği olacaktı

Şekil 1.11: Ortalama bulunması

Örnek: Klavyeden fiyatı girilen bir malın KDV eklenmiş satış fiyatını veren algoritmayı yazınız.



BAŞLA
OKU A
T=A+A/100x18
YAZ T
DUR

T=A+A/100x18 işleminde parantez almaya gerek yoktur. Çarpma ve bölmenin kendi aralarında önceliği olmadığı için A/100x18 işleminin sonucu ile A değeri toplanır.

Şekil 1.12: Satış fiyatı bulunması

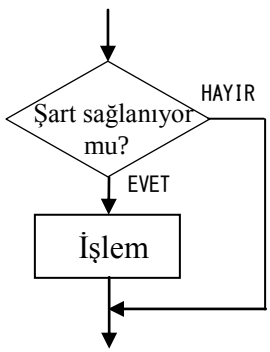
1.4. Programlamada Karar İfadesi

Programlamada karar yapısı "EĞER" ifadesi ile sağlanır. Günlük hayatımızda da çok kullandığımız bu ifade bilgisayar programlarında önemli bir yer teşkil eder. Genellikle programlama dillerinde İngilizce **" if "** olarak kullanılır.

1.4.1. Eğer İfadesinin Tek Durumlu Kullanımı

Su kaynadı mı? → çayı demle
EĞER su kaynadı ise çayı demle

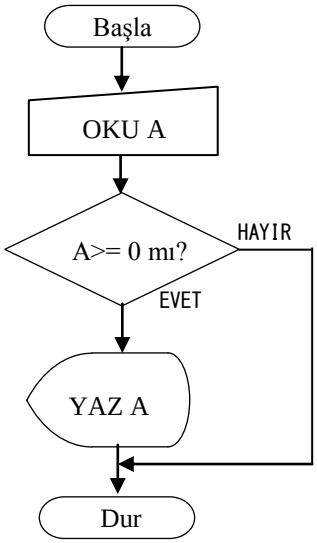
Makine durdu mu? → beyaz çamaşırları topla
EĞER makine durdu ise beyaz çamaşırları topla şeklinde ifade edilebilir.



Eğer ifadesinin tek durumlu kullanımında gerekli şart sağlanıyorsa sadece bir işlem yapılır. Aksi durumda herhangi bir şey yapılmaz.

Şekil 1.13: Tek durumlu karar yapısı

Örnek: Klavyeden girilen pozitif sayıyı ekrana yazdıran algoritmayı tasarlayın.



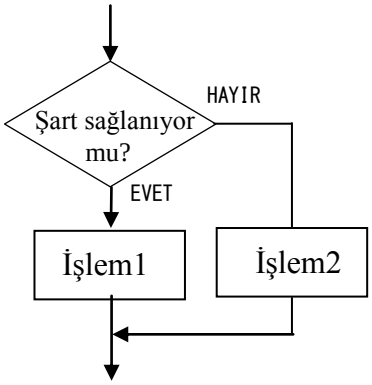

```
BAŞLA
OKU A
EĞER A>=0 İSE
YAZ A
DUR
```

Şekil 1.14: Tek durumlu karar yapısı örneği

1.4.2. Eğer İfadesinin İki Durumlu Kullanımı

Biletin var mı? → Otobüse bin Biletin yoksa → Dolmuşa bin
EĞER biletin var ise otobüse bin DEĞİLSE dolmuşa bin

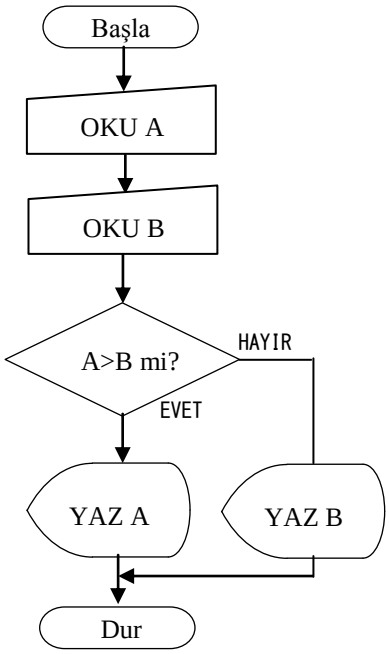
Naktin varsa → Nakit ödeme yap Naktin yoksa → Kredi kartı ile öde
EĞER naktin var ise ödemeyi nakit yap DEĞİLSE kredi kartı ile ödeme yap ile ifade edilebilir.



Eğer ifadesinin iki durumlu kullanımında gerekli şart sağlanıyorsa bir işlem yapılır. Şart sağlanmıyor ise başka bir işlem yapılır.

Şekil 1.15: İki durumlu karar yapısı

Örnek: Klavyeden girilen iki sayıdan hangisinin büyük olduğunu bulan algoritmayı yazınız.

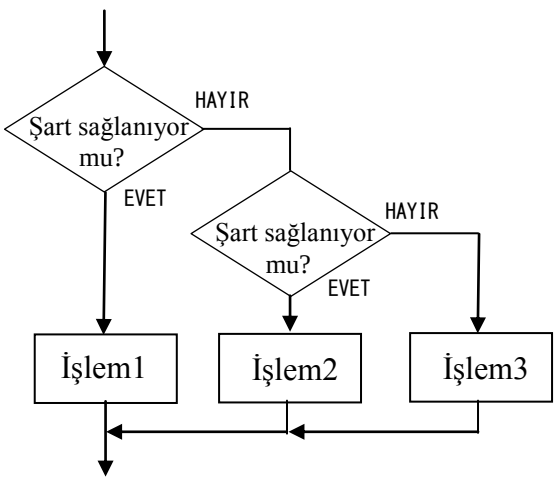



```
BAŞLA
OKU A
OKU B
EĞER A>B İSE
YAZ A
DEĞİLSE
YAZ B
DUR
```

Şekil 1.16: İki durumlu karar yapısı örneği

1.4.3. İç İçe Karar Yapısının Kullanımı

İki durumlu karar yapısında sadece iki olasılık için sonuç alınabiliyordu. İç içe karar yapısında ise bir koşul kendisinden önce gelen koşulun alternatifidir.

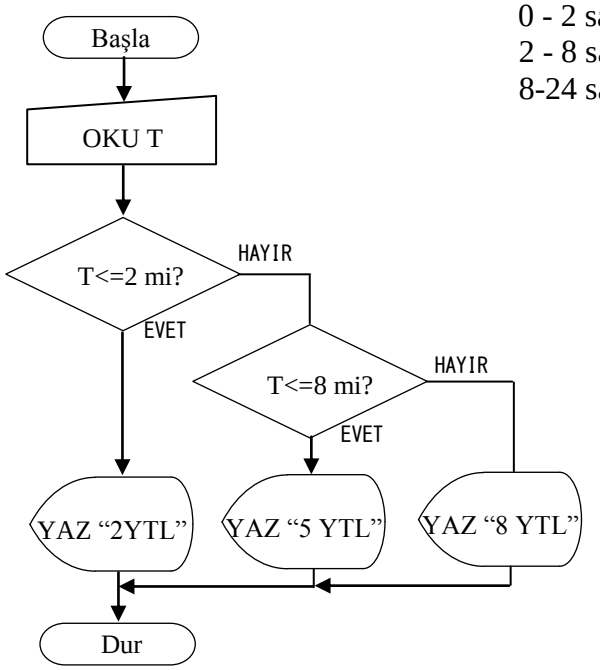



```
EĞER şart gerçekleşiyor ise
      1. işlem
   DEĞİLSE
      EĞER şart gerçekleşiyor ise
            2. işlem
         DEĞİLSE
            3. işlem
      EĞER SONU
EĞER SONU
```

Şekil 1.17: İç içe karar yapısı

Örnek: Aşağıdaki ücret tarifesine göre aracın otopark ücretini hesaplayınız. Araçların otopark içinde en fazla 24 saat kaldığını varsayın.

Zaman değişkeni	T
0 - 2 saat	2 YTL
2 - 8 saat	5 YTL
8-24 saat	8 YTL



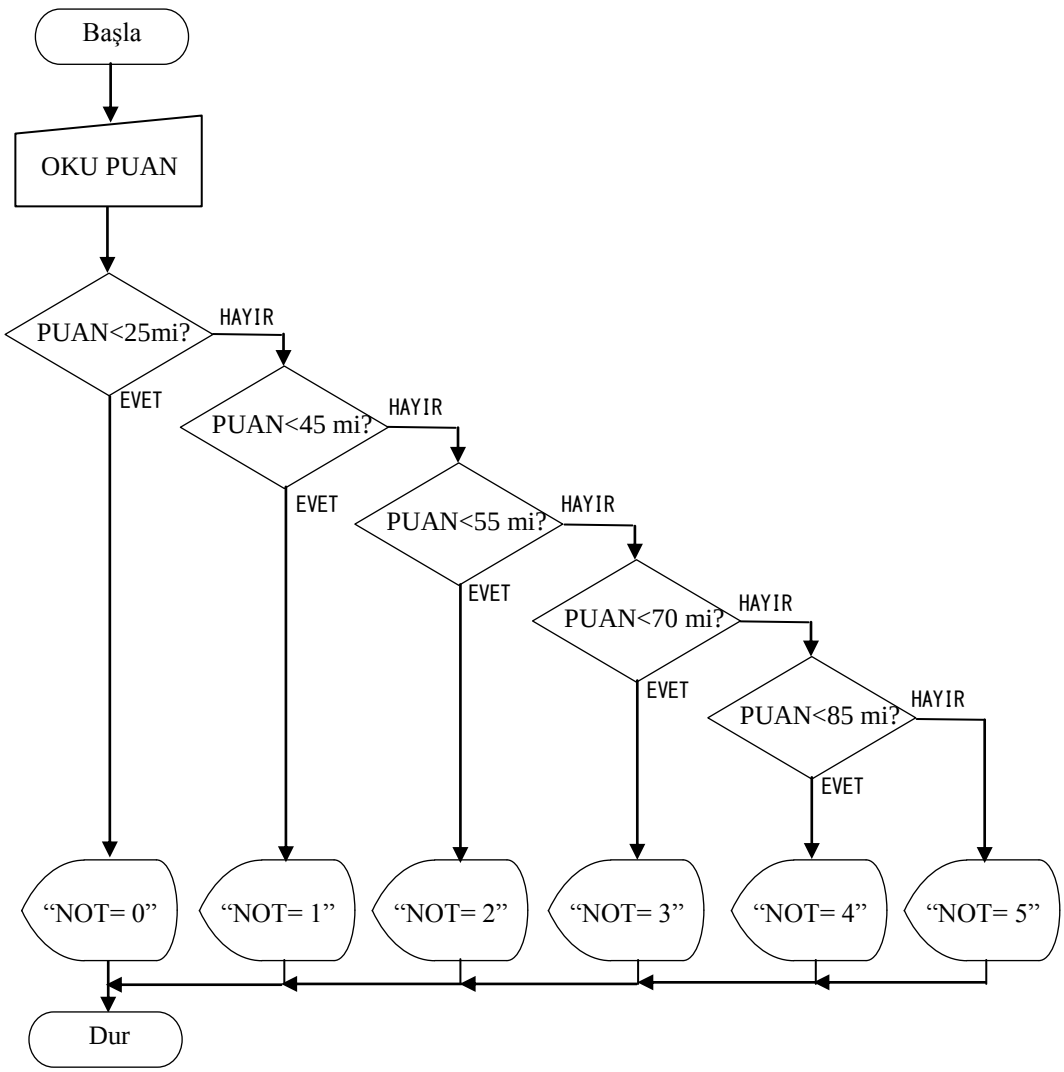

```
BAŞLA
OKU T
EĞER T<=2 İSE
        YAZ "2 YTL"
    DEĞİLSE
        EĞER T<=8 İSE
                YAZ "5 YTL"
            DEĞİLSE
                YAZ"8 YTL"
        EĞER SONU
EĞER SONU
DUR
```

Şekil 1.18: İç içe karar yapısı örneği

Örnek: Klavyeden girilen puan değerine göre öğrenci notunu hesaplayan algoritmayı yazınız.

Not sistemine göre öğrenci puanı

0- 24 ise 0 55- 69 ise 3
25- 44 ise 1 70- 84 ise 4
45- 54 ise 2 85- 100 ise 5 olacaktır.



Şekil 1.19: İç içe karar yapısı örneği

1.5. Sayaç Mantığı

Daha önce incelediğimiz algoritmalarda değişkenler belirleniyor, programa yapılan girdiler ile işlemler gerçekleştiriliyordu. Bu işlemler program işletimi sırasında birer kez yapılıyor ve program sonlanıyordu. Programlamada işlemlerin arka arkaya yapılabilmesi için sayaç mantığı kullanılır.

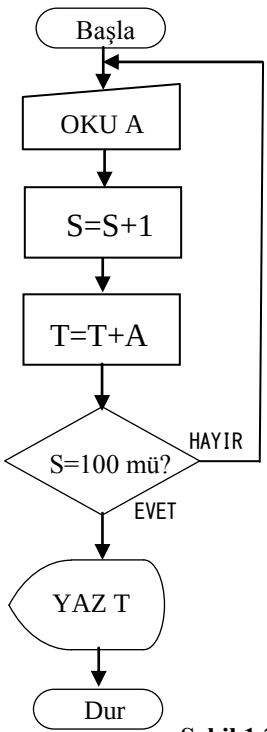
Şimdi bu mantığı bir örnek ile açıklayalım.

Örnek: Klavyeden girilen 100 sayının toplamını bulan ve sonucu ekrana yazdıran algoritmayı tasarlayın.

100 sayının toplamının bulunabilmesi için normal olarak 100 adet değişkenin belirlenmesi ve bunların toplanması gerekir.

sayı1
sayı2
sayı3
. .
. .
sayı99
sayı100

Böyle bir değişken yapısı programın tasarlanması ve işletimi açısından çok hantaldır.
Bu durum 100 kova ile dolan bir su tankı için 100 ayrı kova kullanmaya benzer.
Burada bilinmesi gereken 1 kovanın kaç defa kullanılacağıdır.



Değişkenler

A (Sayı)→ A değişkeni ile klavyeden her defasında değişik bir sayı alabiliriz. Bu değişkenin değeri sürekli değişir çünkü yeni bir sayı aldığında eski değeri siler.

S(Sayaç) → Başlangıçta değeri 0 olan sayaç değişkeni klavyeden sayı girilirken 1 değer artar. Bu durum S=S+1 işlemi ile sağlanır. Buradaki işlem matematiksel açıdan yanlış olsa bile sayaç işletimi için çok önemlidir.

T(Toplam sayı) → Toplam sayı klavyeden girilen sayıların toplamını saklayan değişkendir. Bu değişken de sayaç değeri her arttığında T=T+1 işlemine göre yeni bir değer alır

Şekil 1.20: Klavyeden girilen sayıların toplanması

Şimdi programın işletimi sırasında dikkat edilmesi gereken bazı noktaları kısaca açıklayalım.

OKU A

Klavyeden sayının alınma işlemidir. Bu sayı istenilen değerde bir tam sayı olabilir.

S=S+1

Sayaç değişkeninin başlangıç değeri 0 dır. Bu durumda eşitliğin sağ tarafındaki değer eşitliğin sol tarafına atanır.

Sayacın ilk değeri 0 ise $S = 0 + 1$ S = 1 olur.

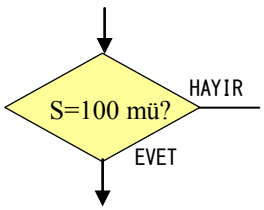
Sayacın yeni değeri 1 ise $S = 1 + 1$ S = 2 olur.

T=T+A

Toplam değişkeninin başlangıç değeri 0'dır. Bu durumda eşitliğin sağ tarafındaki değer eşitliğin sol tarafına atanır.

Klavyeden girilen 1.sayı 4 ise $T = 0 + 4$ T = 4 olur.

Klavyeden girilen 2. sayı 16 ise $T = 4 + 16$ T = 20 olur.

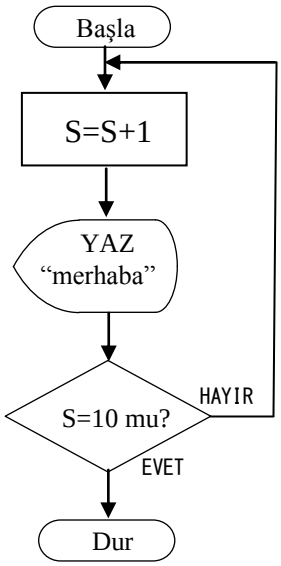


Sayacın kontrolü EĞER koşulu ile sağlanır. Sayaç değeri ilk durumdan itibaren her artışında EĞER koşulu ile sorgulanır. S=100 değil ise klavyeden yeni giriş yapılması sağlanır.S=100 olduğunda 100 sayının toplam sonucu alınmıştır.

Programa rastgele değerler verilerek çalışması incelenebilir.Bu yöntem programların test edilmesinde çok önemlidir.

1.BAŞLA
2.OKU A
3.T=T+A
4.S=S+1
5.EĞER S<>100 ise GİT2
6.YAZ T
7.DUR

Örnek : Ekrana 10 defa "merhaba" yazdıran algoritmayı yazınız.



1.BAŞLA
2.S=S+1
3.YAZ "merhaba"
4.EĞER S<>10 İSE
5.GİT 2
6.DUR

Şekil 1.21: Ekrana 10 defa "merhaba" yazılması

Önceki örnekte sayaç, EĞER karar yapısında belirlenen değere kadar sayıyor, sayma işlemi sona erdiğinde ekrana program sonucu yazılıyordu. Yukarıdaki algoritmada ise S sayaç değeri S=S+1 işlemi ile artmaktadır. Sayaç değeri **her artışında** ekrana "merhaba" yazılır. Sayaç 10 değerine ulaştığında ise yazma işlemi durur.

No	A	S=S+1	T=T+A
1	4	1=(0+1)	4=(0+4)
2	8	2=(1+1)	12=(4+8)
3	5	3=(2+1)	17=(12+5)
4	3	4=(3+1)	20=(17+3)
5	7	5=(4+1)	27=(20+7)
6	9	6=(5+1)	36=(27+9)
7	11	7=(6+1)	47=(36+11)
8	1	8=(7+1)	48=(47+1)
9	16	9=(8+1)	64=(48+16)

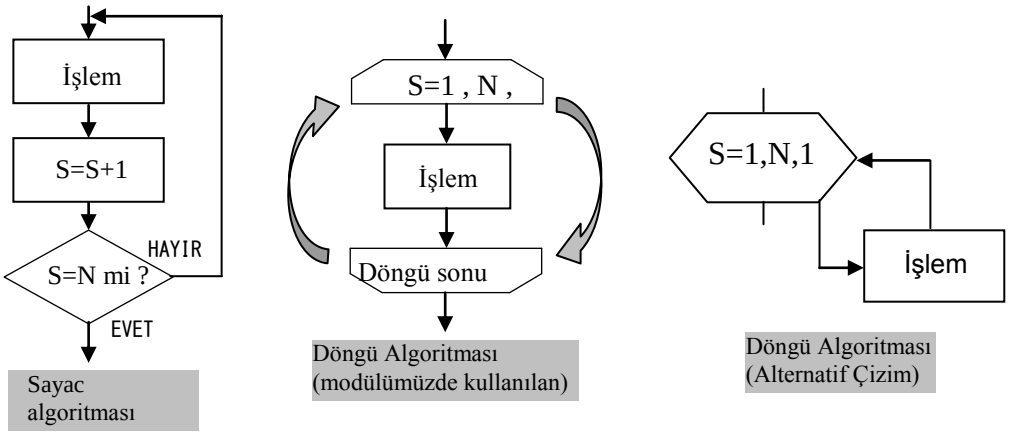
1.6. Döngüler

Döngüler, sayaç mantığı ile yapılan işlemleri otomatik olarak yapan yapılardır.

1.6.1. Temel Döngü Yapısı

Önceki konuda sayaç mantığını öğrendik. Sayaç yapısında bir başlangıç değeri, artış değeri ve koşul vardır. Sayma işlemi başlangıç değerinden başlar ve EĞER karar yapısında belirlenmiş koşul ile sorgulanır. Sayaç artışında koşul sağlanmamış ise program akışı GİT komutu ile başlangıç değerine gider.

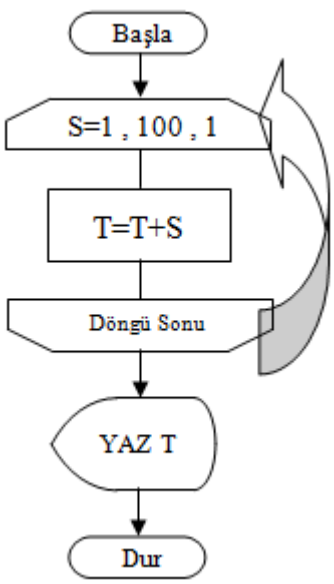
Döngülerde de bir başlangıç sayısı, artış değeri ve koşul vardır.



Şekil 1.22 Sayaç ve Algoritmaları

Döngü kullanımında S=1, N, 1 ifadesi **"S değişkenini 1 değerinden N değerine kadar 1 er arttır"** anlamına gelmektedir.

Örnek : 1` den 100`e kadar olan sayıların toplamını hesaplayan algoritmayı yazınız.



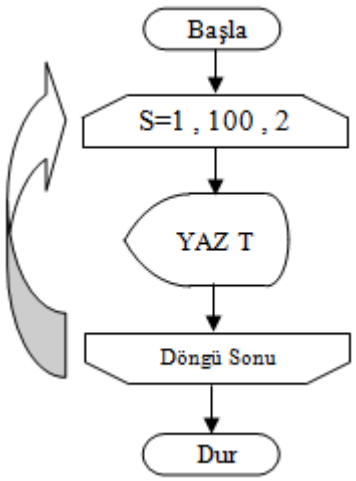

```
BAŞLA
DÖNGÜ S=1,100,1
T=T+S
DÖNGÜ SONU
YAZ T
DUR
```

Döngü içinde başlangıç değeri S=1dir.Sayma işlemi 100 değerinde bitecek ve S değişkeni döngünün her dönüşünde 1 artacaktır.
Program komutlarındaki
DÖNGÜ

DÖNGÜ SONU
ifadesi 1 den 100 e kadar olan toplama işleminin döngü sayma değeri kadar süreceğini belirtmektedir.

Şekil 1.23: Döngü örneği

Örnek : 1` den 100`e kadar olan tek sayıları gösteren algoritmayı yazınız.



BAŞLA
DÖNGÜ S=1,100,2
YAZ T
DÖNGÜ SONU
DUR

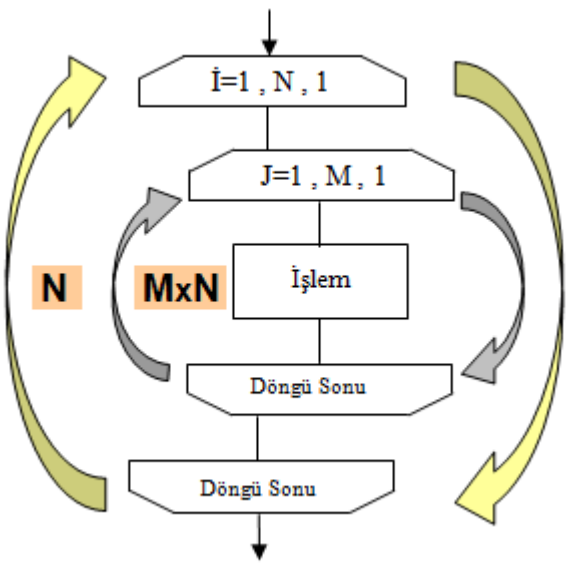
1 den 100 değerine kadar olan tek sayıları göstermek için döngü değişkeninin başlangıç değeri 1, bitiş değeri 100, artış değeri ise 2 olmalıdır.

Şekil 1.24: Döngü örneği

1.6.2. İç İçe Döngüler

Daha önce öğrendiğimiz temel döngü yapısında yapılacak işlem başlangıç ve koşul değerlerine göre otomatik olarak tekrar ediliyordu.

Eğer döngü içindeki bir işlemin alt işlemleri var ise bu durumda iç içe döngü yapısı kullanılır. Bu yapı bilgisayar programcılığında sıkça kullanılan bir yöntemdir.



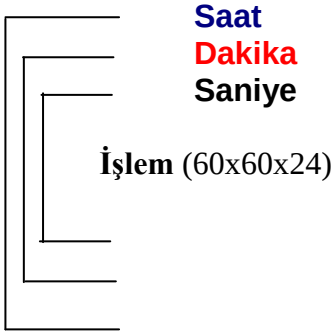
İç içe döngü yapısında iki veya daha fazla döngü bulunur.
Açıklamalarımızı konunun daha iyi anlaşılması açısından 2 döngülü yapı üzerinde yapacağız.

Şekilde görülen birbirinin içine girmiş döngü yapısında dış döngü değeri 1 arttığında iç döngü N sayısına kadar dönmektedir. Bu durumda dış döngünün N kadar dönmesi iç döngü içindeki işlemin **M x N** sayısı kadar yapılacağını gösterir.

Şekil 1.25: Döngü örneği

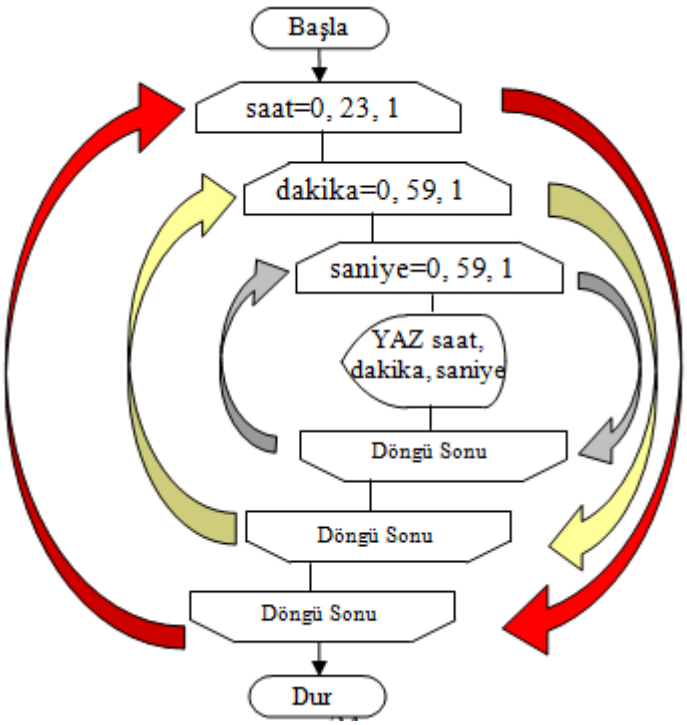
İç içe döngü yapısını şu örnek ile açıklayabiliriz;

1 günlük zaman dilimini dikkate alırsak 60 saniye sonunda 1 dakika, 60 dakika sonunda ise 1 saat geçmektedir.24 saat sonunda gün tamamlamaktadır. Bu durumu iç içe döngü yapısına uyarlarsak saniye en içteki döngü saat ise en dıştaki döngüdür.



Örneğimiz 3 döngülü bir yapıdır. Bir işlemin 1 saniye sürdüğünü varsayalım.

- En içteki saniye döngüsü 0-59 arasında 60 defa döner. Bu durumda dakika döngüsü 1 olur.
- Dakika döngüsü 0- 59 arasında 60 defa döndüğünde ise 1 saatlik zaman dilimi elde edilir.
- Saat döngüsü 24 defa döndüğünde yapılan işlem 60x60x24 =84600 defa tekrar etmiştir.



Şekil 1.26: İç içe döngü örneği

```
BAŞLA
DÖNGÜ SAAT 0,23,1
      DÖNGÜ DAKİKA 0,59,1
          DÖNGÜ 0,59,1
                YAZ SAAT,DAKİKA,SANİYE
          DÖNGÜ SONU
      DÖNGÜ SONU
DÖNGÜ SONU
DUR
```

Örnek : 1-9 arası sayıların çarpım tablosunu gösteren algoritmayı yazınız.

```
BAŞLA
DÖNGÜ İ=1, 9, 1
      DÖNGÜ J=1, 9, 1
                YAZ “Çarpım=”İ x J
          DÖNGÜ SONU
      DÖNGÜ SONU
DUR
```

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

1. Aşağıdaki formüle göre 0 Fahrenheit'tan 210 Fahrenheit' a kadar ısı değerlerinin Celsius karşılıklarını hesaplayan programın algoritmasını çiziniz.

$$C = \frac{(F - 32) \times 5}{9}$$

2. 20'den 50'ye kadar olan sayıların toplamını bulan programın akış diyagramını yazınız.

3. İki tamsayının çarpma işlemini sadece toplama işlemi kullanarak yapan programın akış diyagramını çiziniz.

4. İki tamsayının bölme işlemini sadece çıkarma işlemi kullanarak yapan programın akış diyagramını çiziniz. Bölüm ve kalanın ne olduğu bulunacak.

İşlem Basamakları	Öneriler
➢ Problemi tespit ediniz.	
➢ Problem ile ilgili verileri belirleyiniz.	➢ Verilen bilgileri ve istenenleri not ediniz.
➢ Çözüm için kullanılacak değişkenleri belirleyiniz.	➢ Değişken isimlendirmesini yaparken mümkün oldukça temsil ettiği değeri hatırlatacak kısa isimler vermeye çalışınız.
➢ Problemi adım adım kağıda yazınız.	➢ Adımları belirlerken gereksiz adımlardan kaçınınız. ➢ Her adımı sadece bir işlemi temsil edecek şekilde hazırlayınız.
➢ Akış diyagramını çiziniz.	➢ Her algoritmanın başının ve sonunun belli olması gerektiğini unutmayınız. ➢ Algoritma adımlarında problemin çözümünü etkileyecek her hangi bir belirsizlik olmamalıdır. Dikkat ediniz. ➢ Algoritma içinde döngü kullanılmış ise başlangıç, bitiş ve artım miktarını yazmayı unutmayınız. ➢ Algoritma hazırlanırken eğer bir karar yapısı varsa; yapılan işlem sonucunda her türlü alternatifi değerlendiriniz. Bir sorunun cevabının evet olabileceği gibi hayır olabileceği de unutmayınız.
➢ Algoritmayı kontrol ediniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet** kazanamadığınız becerileri **Hayır** kutucuğuna **(X)** işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Problemi doğru tespit ettiniz mi?		
2. Problemle ilgili verileri belirlediniz mi?		
3. Kullanılacak değişkenleri belirlediniz mi?		
4. Problemi adım adım yazdınız mı?		
5. Akış diyagramını çizdiniz mi?		
6. Akış diyagramına bağlı olarak algoritmayı oluşturdunuz mu?		
7. Oluşturduğunuz algoritmayı kontrol ettiniz mi?		
8. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
9. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet sonunda kazandıklarınızı aşağıdaki soruları cevaplandırarak ölçünüz.

1. Aşağıdaki işlemin bilgisayar programındaki karşılığını tamamlayınız.

$$R = \frac{R1 \times R2}{R1 + R2} \rightarrow R = R1 * R2 / .....$$

2. Aşağıdaki aktarma işlemlerinin sonuçlarını tabloda gösteriniz.

A=1

B=1

C=A+B

D=C+B

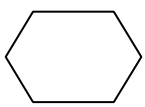
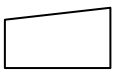
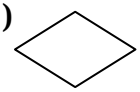
E=D+A

C	D	E

3. Tabanı ve yüksekliği verilen bir üçgenin alanını bulan program için gereken ifade ile boşluğu doldurunuz.

BAŞLA
OKU TABAN
OKU YÜKSEKLİK
ALAN = _____
YAZ ALAN
DUR

4. Eğer ifadesi kullanımına akış diyagramında karşılık gelen sembol hangisidir?

A)  **B)**  **C)**  **D)**

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Bilgisayar programı yazım aracını hatasız bir şekilde kuracak ve programı derleyebileceksiniz.

ARAŞTIRMA

- Bilgisayar programlama dilleri hakkında araştırma yapınız.
- Derleyiciler hakkında araştırma yapınız.

2. BİLGİSAYAR PROGRAMLAMA

Bilgisayar iki bölümden oluşur. Bunlardan birincisi bilgisayarın fiziksel görünüşünü oluşturan klavye, fare, ekran vb. Parçalarının bütünüdür. Bilgisayarın bu fiziksel parçaların bütünüdür. Bilgisayarın bu fiziksel parçalardan oluşan kısmına donanım denir. İkinci kısmı ise yazılımdır. Bu bölüm bilgisayara işlerlik kazandıran, kullanıcının bilgisayara birtakım işler yaptırabilmesini sağlayan bölümdür.

2.1. Yazılım Kavramı

Bilgisayarın fiziksel görünüşüne ek olarak ona yaptırmak istediklerimiz için birtakım emirler verebilmemiz gerekir. Bilgisayara yaptırmak istediğimiz işler için vereceğimiz emirlere **komut** denir. Örneğin bilgisayara verebileceğimiz kaydet, sil, yazıcıya gönder gibi emirler birer komuttur. Bu komutlar bilgisayar ile kullanıcı arasında ilişki kuran özel bir dil görevi yapar. Bilgisayara istenilen işleri yaptırmak için kullanılan komutlar zincirine **program** denir. Bilgisayara yaptırılacak işler için verilen komutlar ile bu komut zincirinden oluşan bilgisayar programları birer yazılımdır. Kısaca, bir bilgisayarın fiziksel görünümüne işlerlik kazandıran, insan çabası sonunda meydana getirilen programlar ve fonksiyonlara **yazılım** adı verilir.

2.1.1. Bilimsel Ve Mühendislik Yazılımları

Bilimsel ve mühendislik konularındaki problemlerin çözülmesinde kullanılan programlardır. Bu tür programlarda veri miktarı göreli olarak düşüktür ancak matematiksel ve istatistiksel algoritmalar yoğun olarak kullanılabilir. Tamamen hesaplama ağırlıklı işlemler içerir. Elektronik devrelerin çözümünü yapan programları, istatistik analiz paketlerini bu tür programlara örnek olarak verebiliriz.

2.1.2. Mesleki Yazılımlar

Veritabanı ağırlıklı yazılımlardır. Genel olarak verilerin yaratılması, işlenmesi ve dosyalarda saklanması ile ilgilidir. Bu tür programlara örnek olarak stok kontrol programları, müşteri takip programları, muhasebe programlarını verebiliriz.

2.1.3. Yapay Zeka Yazılımları

İnsan davranışlarını taklit etmeyi amaçlayan yazılımlardır. Örnek olarak robot yazılımları, satranç gibi zeka oyunlarını oynatan programlar vs. verilebilir.

2.1.4. Görüntüsel Yazılımlar

Görüntüsel işlemlerin ve algoritmaların çok yoğun olarak kullanıldığı programlardır. Örnek olarak oyun ve animasyon yazılımlarını verebiliriz. Bu yazılımlar ağırlıklı olarak bilgisayarın grafik arabirimini kullanırlar.

2.1.5. Sistem Yazılımları

Bilgisayarın elektronik yapısını yöneten yazılımlardır. Derleyiciler, haberleşme programları, işletim sistemi birer sistem yazılımıdır. Örneğin text editörü de bir sistem yazılımıdır. Uygulama programlarına göre daha düşük seviyeli işlem yaparlar.

İki tabanlı (binary) ve onaltı tabanlı (heksadesimal) sayı sistemleri bilgisayar programlamaya yeni başlayanlar için öğrenilmesi gerekli temel konuların başında gelir.

2.2. Sayı Sistemleri

Günlük yaşantımızda 35 YTL, 4 kg veya 17 haziran ifadelerini kullandığımızda herkes ne söylediğimizi anlayabilir. Buradaki 35, 4 ve 17 desimal sayı sistemine aittir.

2.2.1. İki Tabanlı (Binary) Sayı Sistemi

Sayısal elektronik sistemler bize yabancı bir sayı sistemi kullanır. Bu sayı sistemine ikilik sayı sistemi (binary) denir. Bilgisayarlar ve diğer mikroişlemci tabanlı sistemler onaltılık (heksadesimal) ve sekizlik (oktal) sayı sistemleriyle çalışır.

Desimal sayı sistemi 0,1,2,3,4,5,6,7,8,9 rakamlarından oluşur.Bu nedenle onluk veya on tabanlı sayı sistemi olarak adlandırılır.Binary sayı sisteminde sadece 0 ve 1 rakamları olduğu için ikilik veya iki tabanlı sayı sistemi olarak adlandırılır.

Şekil 2.1'da daireler madeni parayı ifade etmektedir. Bu tabloda; madeni paraların miktarları desimal sayı sistemi için sol sütunda ve binary sayı sistemi için sağ sütunda

sembolize edilmiştir. Örneğin dokuz madeni paranın olduğu satırda bu paraların miktarını desimal sayı sistemine göre belirtmek istediğimizde 9 sembolünü kullanmamız gerekir. Aynı miktarı binary sayı sistemi ile ifade etmek için 1001 sembolünü kullanmalıyız. 0 ve 1 sayıları desimal ve binary sayı sisteminde kullanılan ortak sembollerdir.

Desimal	Binary
○ 0	0
● 1	1
●● 2	10
●●● 3	11
●●●● 4	100
●●●●● 5	101
●●●●●● 6	110
●●●●●●● 7	111
●●●●●●●● 8	1000
●●●●●●●●● 9	1001

Şekil 2.1: Sayı sembolleri

İki tabanlı sayı sisteminde 10 sayısını ’bir sıfır’ şeklinde, 111 sayısını ’bir bir bir’, 1001 sayısını ise ‘bir sıfır sıfır bir’ şeklinde söylememiz gerekir.

İki tabanlı sayı sistemindeki, her sembol **bit** (basamak) olarak adlandırılır. Her bit **0** ve **1** olmak üzere iki farklı değer alabilir. Buna göre **n** bitlik bir sayı $\mathbf{2^n}$ değişik şekilde oluşturulabilir. **0** ve **1** sayıları elektrik devrelerinde sinyal var ya da yok anlamında kullanıldığı gibi, bir lambanın yanıp yanmadığını veya bir anahtarın kapalı mı yoksa açık mı olduğunu da gösterir.

Örnek: Aşağıda 4 bitlik sayılar verilmiştir. Bu sayılardaki her bitin bir lamba olduğunu düşünerek ledlerin yanıp yanmayacağını belirleyiniz.

a. 1010 b. 1000 c. 1111

Çözüm :

Binary sayılar	Lambalar
1010	●○●○
1000	●○○○
1111	●●●●

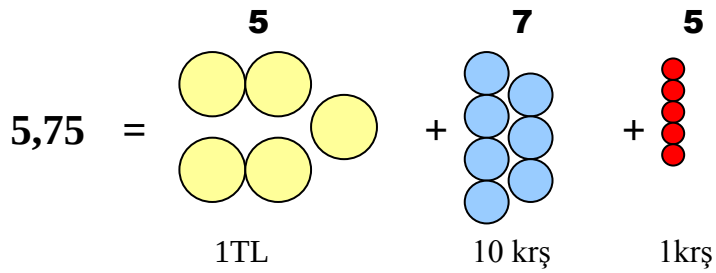
Şekil 2.2: Lambalar kullanılarak binary sayıların gösterimi

Örnek: Binary sayı sisteminde 4 bit, 8 bit ve 16 bit kullanılarak kaç farklı sayı yazılabilir?

Çözüm : 4 bitlik $2^4 = 16$
8 bitlik $2^8 = 256$
16 bitlik $2^{16} = 65536$ değişik sayı yazılabilir.

2.2.2. Basamak Değerleri

İki tabanlı sayı sistemlerinde basamak sistemini on tabanlı sisteme göre açıklayabiliriz. Örneğin alışverişe çıktınız ve aldığınız malın tutarı 5,75 TL olsun. Bu ücreti satıcıya aşağıdaki şekilde olduğu gibi 5 adet 1 TL, 7 adet 10 kuruş ve 5 adet 1 kuruş olarak ödeyebiliriz.



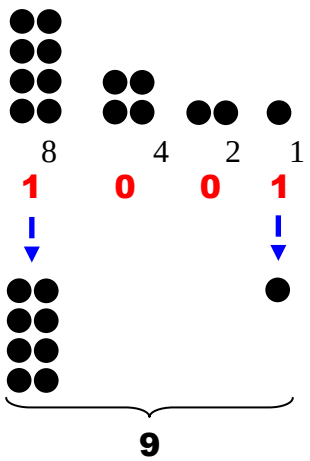
Şekil 2.3: Basamak sistemi

648 sayısını yukarıdaki örneğe göre basamaklarına ayırırsak, soldaki 6 rakamı yüzler basamağını, 4 rakamı onlar, 8 rakamı da birler basamağını gösterir.

648 = 600 + 40 + 8

İki tabanlı sayı sistemini de basamaklarına ayırabiliriz.

Örneğin 1001 (bir sıfır sıfır bir olarak okunur.)sayısını düşünelim.



Onluk sayı sisteminde basamaklar sağdan sola 10'nun katları şeklinde artıyordu.
İki tabanlı sayı sisteminde basamak değerleri sağdan sola 2'nin katları şeklinde artar.

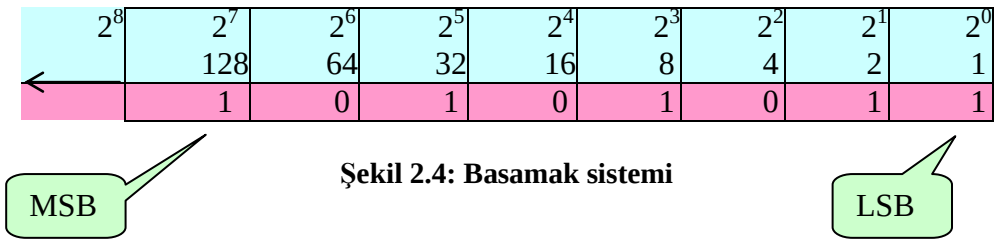
- 1001 sayısında 1. basamağında "1" değeri vardır.
- 2 ve 4. basamakta "0" olduğu için bu basamakların sayısal değeri yoktur.
- 8. basamakta "1" değeri olduğu için 1001 sayısının desimal karşılığı 8+1=9 dur.

Binary (iki tabanlı) sayı sisteminde taban 10 yerine 2 olduğu için basamak değerleri 2^0=1, 2^1=2, 2^2=4, 2^3=8, 2^4=16, 2^5=32 ... şeklinde ifade edilir.

Örnek olarak; $(00101011)_2$ sayısının desimal sistemde $(43)_{10}$ a eşit olduğunu şu şekilde kanıtlayabiliriz:

$=0x128+0x64+1x32+0x16+1x8+0x4+1x2+1x1 =(0+0+32+0+8+0+2+1) =(43)_{10}$

En soldaki binary rakam '**En Büyük Dereceli Rakam**' (The **M**ost **S**ignificant **B**it-M**SB**), en sağdaki rakam ise '**En Küçük Dereceli Rakam**' (The **L**east **S**ignificant **B**it-**LSB**) olarak adlandırılır. Bu ifade **Şekil 2.4'te MSB ve LSB** olarak belirtilmiştir.



Şekil 2.4: Basamak sistemi

Sayının Değeri $= (1\times2^7)+(0\times2^6)+(1\times2^5)+(0\times2^4)+(1\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)$
$= (1\times128)+(0\times64)+(1\times32)+(0\times16)+(1\times8)+(0\times4)+(1\times2)+(1\times1)$
$= 128 + 0 + 32 + 0 + 8 + 0 + 2 + 1$
$= (171)_{10}$

Desima			Binary			
100	10	1	8	4	2	1
		0				0
		1				1
		2			1	0
		3			1	1
		4		1	0	0
		5		1	0	1
		6		1	1	0
		7		1	1	1
		8	1	0	0	0
		9	1	0	0	1
	1	0	1	0	1	0
	1	1	1	0	1	1
	1	2	1	1	0	0
	1	3	1	1	0	1
	1	4	1	1	1	0

Şekil 2.5: Binary sayı sistemi

2.2.3. Onaltı Tabanlı (Heksadesimal) Sayı Sistemi

Binary sayı sistemi bilgisayarın anladığı tek sayı sistemidir. Bilgisayardan girdiğiniz tüm yazı, sayı ve işlemler binary sayıya çevrilerek bilgisayar tarafından algılanır. Fakat binary sayı sisteminde yalnızca 2 rakam olduğu için büyük sayıları ifade etmek oldukça fazla rakamla mümkün olur. Örneğin desimal sayı olan 202 sayısını 3 rakam kullanarak ifade edebilirken aynı sayıyı binary sayı sisteminde 11001010 şeklinde yazarız ki bu bizim 8 rakam kullandığımızı gösterir.

Bilgisayar üreticileri bu sorunu heksadesimal sayı sistemini geliştirerek çözmüşlerdir. Bu sayı sisteminde sayılar daha az rakam kullanılarak ifade edilebilmektedir. Ayrıca bu sayı sisteminin ayrı bir üstünlüğü de binary sayıya geçiş ve binary sayıdan heksadesimal sayıya geçişin kolay olmasıdır. Heksadesimal sayı sisteminde 16 sembol kullanılır. Bunlar 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F dir.

'A' harfi 10 sayısına, 'B' harfi 11 sayısına , 'C' harfi 12 sayısına, 'D' harfi 13 sayısına, 'E' harfi 14 sayısına ve 'F' harfi de 15 sayısına karşılık gelir. Bunun yanında heksadesimal sayıyı yazarken diğer sistemlerde de olduğu gibi tabanı belirtmek zorundayız. Bu belirtme örneğin; **$(1A)_{16}$** şeklinde olabileceği gibi **1Ah** şeklinde de olabilir.

Heksadesimal sayılar 4'er bit şeklinde ayrılırlar. Örneğin 0000 0101 binary sayıyı heksadesimal olarak ifade ederken ilk dört bitin değeri sıfır olsa dahi **05h** olarak yazmak zorundayız. Örnek olarak heksadesimalde $(F)_{16}$ sayısı, binary sistemde $(1111)_2$ olarak gösterilir. Ya da $(9)_{16}$ sayısı $(1001)_2$ dir.

Diğer bir örnek ise $(4C)_{16}$ sayısı 8 bit binary sayıyı ifade eder. Binary yazılımı ise $(0100\ 1100)_2$ dir. Heksadesimal sayı sistemi mikroişlemci temelli uygulamalarda binary sayı sistemindeki 8, 16, 32, 64 biti gösterebilmek için sıklıkla kullanılır.

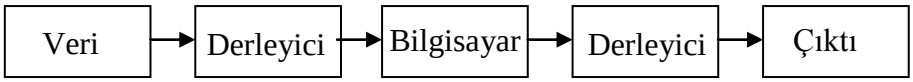
2.3. Programlama Dilleri

Bir dil, yalnızca o dilden anlayan insanlar ile iletişim kurabilmemizi sağlayan bir araçtır. Ana dili bizimki ile yanı olmayan insanlar ile anlaşabilmek için onların dillerini öğrenmeye çaba gösteririz. Bir bilgisayar ile anlaşabilmek için de bilgisayarın kabul edebileceği dili öğrenmek gerekir. Bilgisayara ancak anladığı dilde komutlar verilirse istenenler yaptırılabilir.

Bilgisayarın kullanabileceği ve anlayabileceği dile **makine dili** denir. Makine dili "01010101" gibi iki tabanlı sayılardan oluşur. Bilgisayarın kullanılmaya başlandığı ilk yıllarda, programlar makine dilinde yazılarak hazırlanmaktaydı.

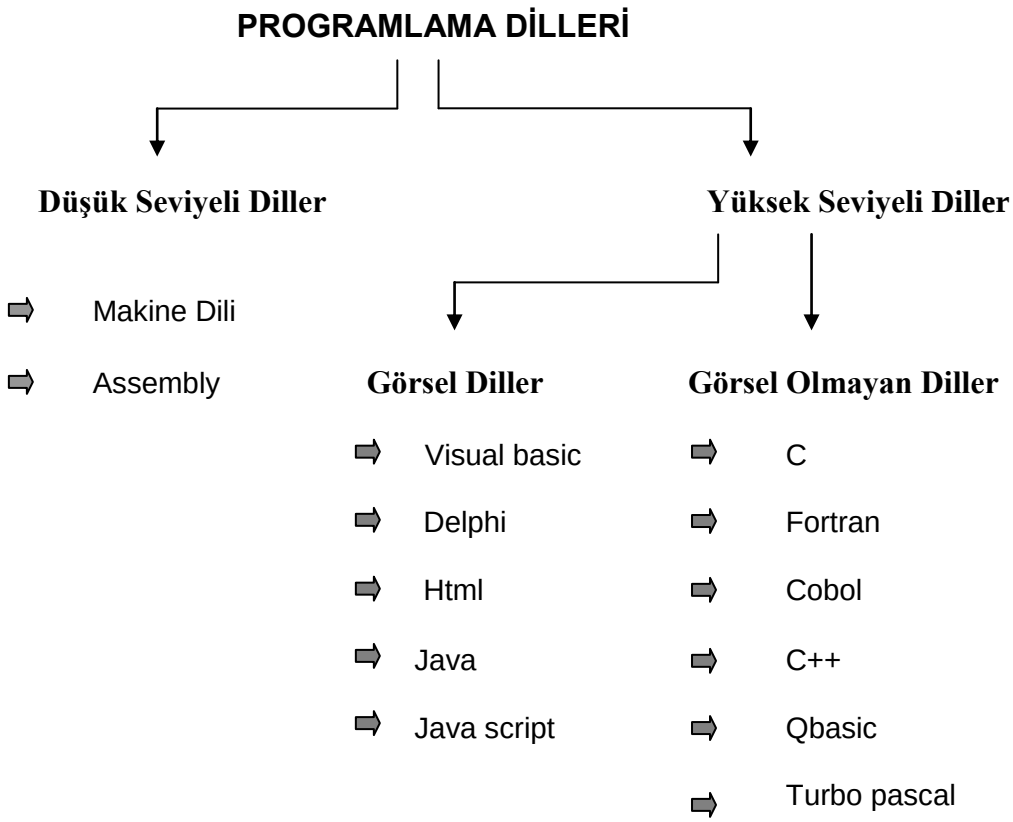
Makine dilinde program yazmanın zorluğu insanları bir başka yolla program yazabilme arayışına götürmüştür. Bir süre sonra da kolay öğrenilen ve problemleri daha kolay çözmeye olanak veren **yüksek seviyeli** programlama dilleri geliştirilmiştir. Bunun yanında, programlama dilleriyle yazılan programları bilgisayarın anlayacağı hale çevirecek

derleyiciler de ortaya çıkmıştır. **Derleyici** (compiler), programlama dilinde yazılan programları makine diline çeviren programdır.



Şekil 2.6: Bilgisayarda veri işleme

Derleyiciler program kodlarını, klavyeden giriş hataları ve dilin programlama komutlarının yanlış kullanımı gibi teknik hatalar için kontrol ederler. Ancak programlardaki yanlış bir veri veya formül gibi hataları kontrol edemezler. Programı mantık hatalarından arındırmak programcının işidir.



Şekil 2.7: Programlama dilleri

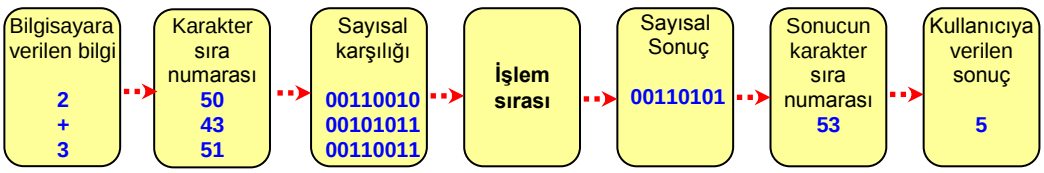
Derleyicisi olan dillerde yazılan program derleyiciden geçirilerek yazım hatası, sayısal hata, komut, sıra hatası vb. gibi hatalar varsa bu hatalar listelenir. Programcı bu hataları gidererek programı yeniden derler. Bu tür programlar ancak bütün olarak derlendikten sonra çalıştırabilir.

Bilgisayara girilen veriler şekli ve tipi ne olursa olsun, önce derleyiciye ulaşarak makine diline çevrilir. Bilgisayar verileri işler. Elde edilen sonuçlar yine derleyicide kullanıcının anlayabileceği bir dile çevrilir. Bilgisayar anlaşılabilir duruma getirilen sonuçları verir. Bu durumu aşağıdaki örnek ile açıklayalım.

Bilgisayar belleğinde tanımlı 256 farklı karakter vardır. Bu karakterlere **ASCII kodu** adı verilir. Bilgisayar bu karakterlerin her birini bir sıra numarasıyla tanır. Örneğin "A" harfinin sıra numarası 65 "a" harfinin sıra numarası ise 97'dir.

Bilgisayara bir karakter (harf, rakam ya da özel işaret) yazıldığında, önce bu karakterlerin sıra numarası bulunur. Bilgisayar, bu sıra numarasının 0 ve 1'lerden oluşan sayısal karşılığını bulur. Bilgi bu şekliyle bilgisayarın anlayabileceği hale gelmiş olur. İşlemler de bu bilgi üzerinden yapılır. Daha sonra çıkan sonuç yine 0 ve 1'lerden oluşan bir sonuçtur. Bilgisayar bu kez 0 ve 1'lerden oluşan bu bilginin sıra numarasını bulur. Bu sıra numarasına karşılık gelen karakteri sonuç olarak verir.

Kullanıcı yalnızca verdiği bilgi ve aldığı sonucu görür. Diğer işlemler ise bilgisayar ve derleyicilerde otomatik olarak yapılır.



Şekil 2.8: Bilgisayarda veri işleme sırası

2.4. C Programlama Dili

C bir sistem programlama dilidir. Sistem Programlama; donanımın yönetilmesi, kontrolü ve denetimi için yazılan, doğrudan donanımla ilişkiye giren programlama şeklidir. Örneğin, işletim sistemleri, derleyiciler, yorumlayıcılar, aygıt sürücüleri, bilgisayarların iletişimine ilişkin programlar, otomasyon programları sistem programlarıdır. Diğer uygulama programlarına destek veren yazılımlar da çoğunlukla sistem programları olarak ele alınırlar.

2.4.1. C Programlama Dili Özellikleri

C'den önce sistem programları assembly dillerle yazılıyordu. C dilinin sistem programlarının yazılmasında hemen hemen alternatifsiz olduğunu söylenebilir. Bugün cep telefonlarından, uçaklardaki bilgisayar sistemlerine kadar her yerde C kodları çalışmaktadır.

C dilinin özelliklerini şu şekilde sıralayabiliriz.

- C algoritmik bir dildir. C'de program yazmak için yalnızca dilin yazım kurallarını ve yapısını bilmek yetmez, genel bir algoritma bilgisi de gerekir.

- C diğer dillerle kıyaslandığında taşınabilirliği çok yüksek olan bir dildir. Çünkü 1989 yılından bu yana genel kabul görmüş standartlara sahiptir. İfade gücü yüksek, okunabilirlik özelliği güçlü bir dildir.
- C çok esnektir. Diğer dillerde olduğu gibi programcıya kısıtlamalar getirmez.
- Güçlü bir dildir. Çok iyi bir biçimde tasarlanmıştır. C'ye ilişkin operatörlerin ve yapıların birçoğu daha sonra başka programlama dilleri tarafından da benimsenmiştir.
- C verimli bir dildir. Seviyesinden dolayı hızlı çalışır. Verimlilik konusunda assembly diller ile rekabet edebilir.
- C doğal bir dildir. C bilgisayar sisteminin biçimiyle uyum içindedir.

2.4.2. C Programının Derlenmesi

Daha önce de bahsedildiği gibi bilgisayarlar sadece makine dilinde yazılan kodları işleyebilirler. Yüksek seviyeli dillerle yazılan programlar makine diline derleyiciler (compiler) vasıtasıyla dönüştürülürler.

C programlama dilinde derleme işlemi şekil: 2.4 de gösterilmiştir. Kaynak kod herhangi bir kelime işlemci programı yardımıyla girilir. (kaynak kodun dosya uzantısı .c olmak zorundadır.) Öncelikle, önişlemci (preprocessor) kaynak kod içindeki # ile başlayan önişlemci komutlarını kontrol eder. Daha sonra bir makroyu atar yada bazı kütüphane dosyalarını okur ve işler. Bundan sonraki adımda derleyici tarafından işlenen dosya, makina diline çevrilir. Makina dili haline dönüşmüş kodun bu haline nesnesel program (object program) adı verilir. Bu adımda nesnesel program, fonksiyonların makina diline çevrilmiş kodunu ve işletim sisteminde işletilecek gerekli bilgileri içermez.

Bağlayıcı (Linker) fonksiyonların makina kodunu, nesnesel program içerisindeki kütüphane adı verilen dosya içerisinde bağlamaya ve bu kodu çalıştırmak için gerekli bilgileri hazırlamaya yarar ve dosyanın son hali "çalıştırılabilir dosya" (**executable program**) olarak adlandırılır.

Temel bir C programı aşağıdaki gibi olacaktır.

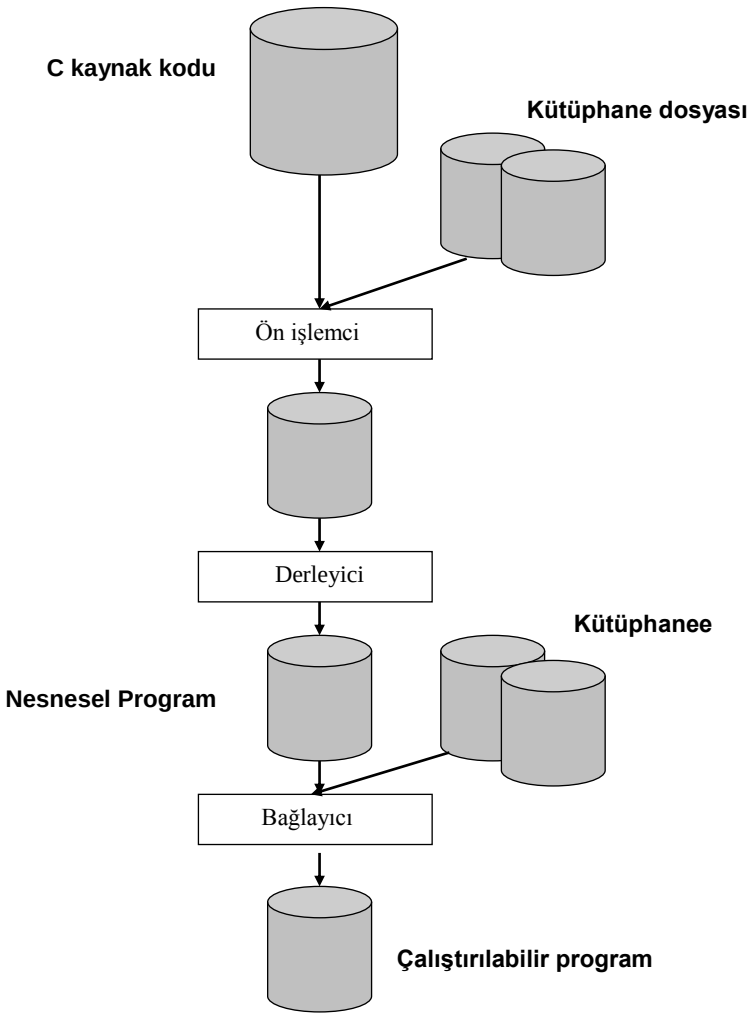
```
/*  ilk C programi */

 #include <stdio.h>

 main()
 {
   printf("Merhaba Dünya!\n");
 }
```

1
2
3
4

- Program içine /* -------- */ şeklinde yazacağınız açıklama satırları derleyici tarafından dikkate alınmaz.
- " # " işareti programda önişlemci kullanılacağını belirtir. Her C programı <stdio.h> ifadesi ile başlar.Standart giriş çıkış (standart input output) anlamına

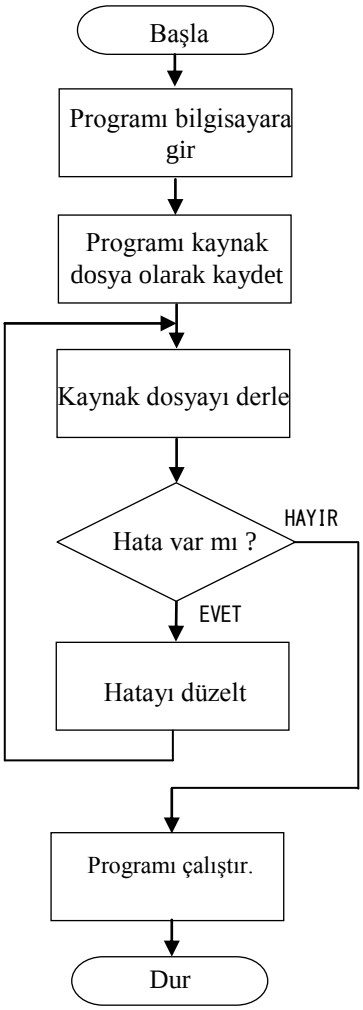
gelen bu ifade temel komutların kullanılabilmesini sağlayan kütüphane dosyasıdır.Başka bir ifade ile **#include<stdio.h>,** derleyiciye stdio.h kütüphanesindeki fonksiyonları program ile birlikte derleme işini yaptırır.

- C programları fonksiyonlardan meydana gelir."main()" (temel) fonksiyonu her programda mutlaka kullanılması gereken bir fonksiyondur. Bu fonksiyon küme parantezi ile başlar ve biter.
- Bilgisayar ekranına çıktı verebilmek için printf komutu kullanılır. Komut içine yazılacak ifade " " karakterleri arasında olmalıdır. Program fonksiyonu (main()) içinde kullanılan her komut; karakteri ile sonlandırılmalıdır. Aksi halde derleyici komutu dikkate almaz. "\n" ifadesi "merhaba dünya" yazdıktan sonra bir alt satıra geçmek için kullanılır.



Şekil 2.9: Derleme İşlemi

Bir C programını derleme işleminin algoritması aşağıdaki şekilde olabilir:



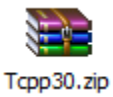
Şekil 2.10: Program derleme işlem sırası

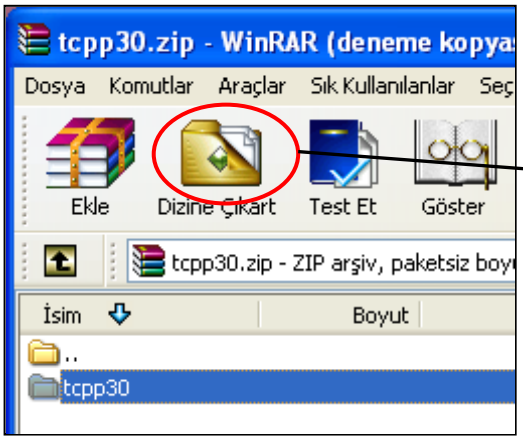
2.4.3. Program editörünün kullanılması

Bir bilgisayar programındaki tüm komutlar ve ifadelerin hepsine "Kaynak Kodu" adı verilir. Kaynak kodu bilgisayarın hafızasına program editörü kullanılarak yazılır. Yazılan program derleme (compile) ve çalıştırma (run) işleminden önce bir diske (disket veya sabit disk vb.) uzantısı ".C" olacak şekilde bir dosya altında saklanır.

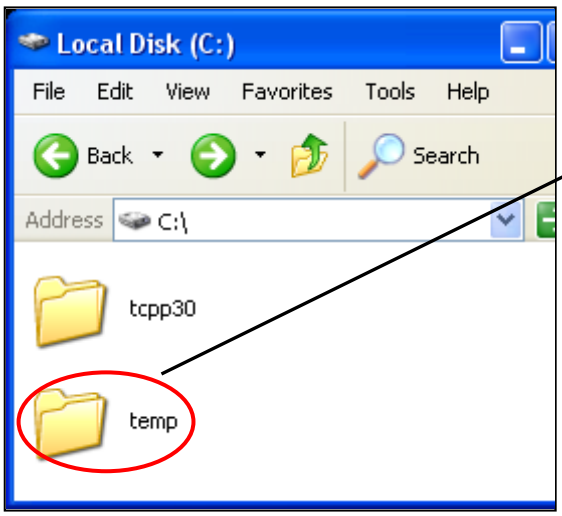
Aşağıda "Turbo C " derleyicisinin kurulumu ve örnek programın derlenmesi açıklanmaktadır.

Derleyici ve program editörünün aynı yapıda kullanıldığı Turbo C internet üzerinden ücretsiz olarak indirilebilir.

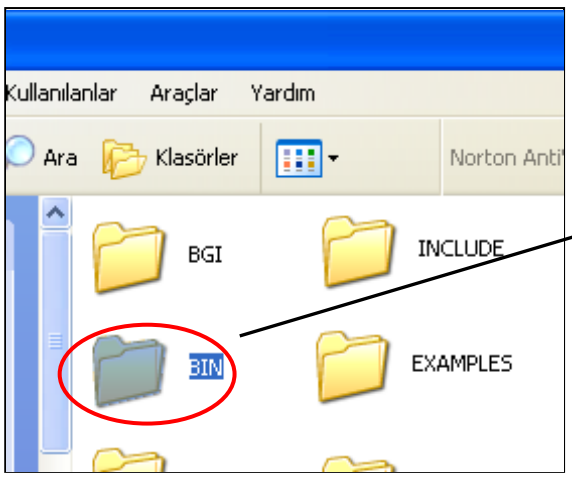




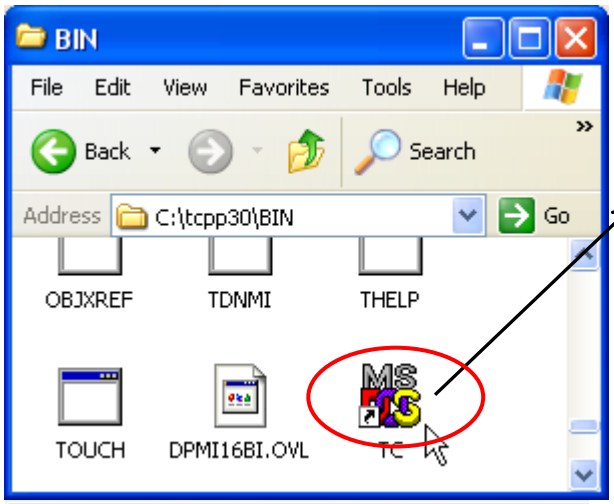
Sıkıştırılmış Tcpp30 dizini bilgisayarda belirlenecek sabit disk içinde açılır.



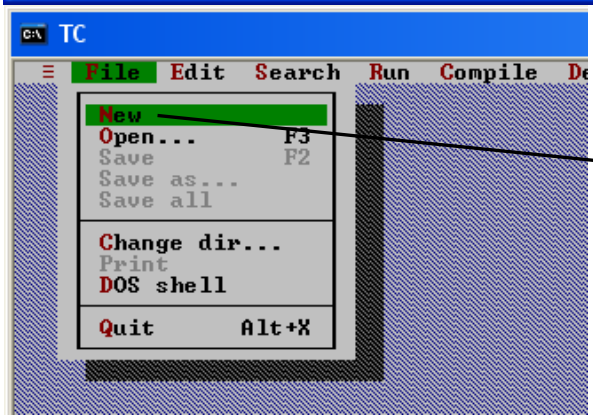
Sabit disk içinde açılan Tcpp30 dizini ile beraber yeni bir dizin oluşturmalısınız. Bu dizinin ismini daha sonra değiştirebilirsiniz.
Yapacağınız programların .exe uzantılı uygulanabilir dosyaları ve nesne dosyaları bu dizin içinde saklanacaktır



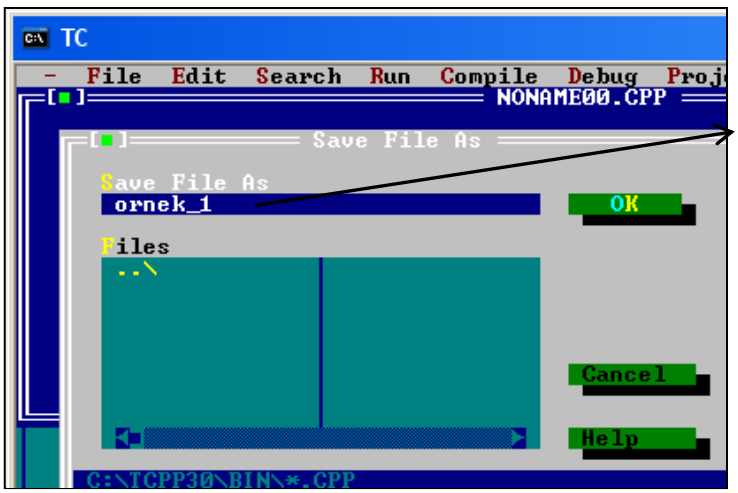
Tcpp30 dizini içindeki BIN dizinine girilir.



BIN dizini içindeki TC isimli MS-DOS sembollü simge tıklanır.

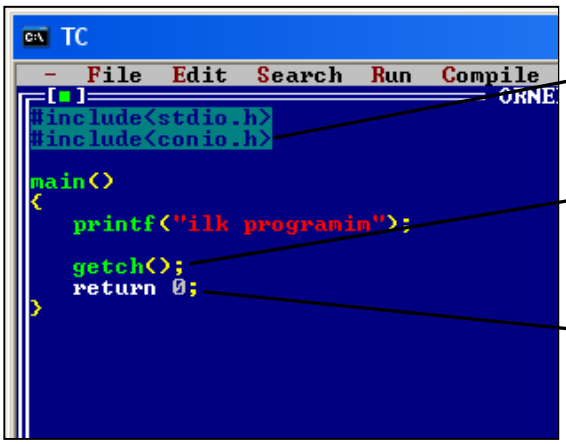


Editör menüsünden yeni bir çalışma sayfası açılır.
file → new



Açılan çalışma sayfasına isim verilir. (ornek_1)
file → save as

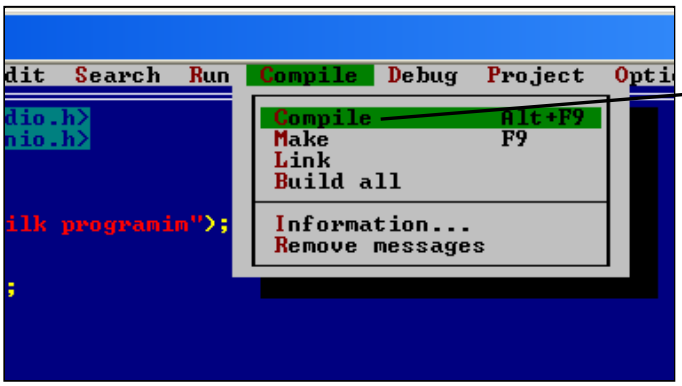
Turbo C editöründe programın ekran görüntüsü ve fonksiyon özellikleri için aşağıdaki ifadelerin programlarınıza eklenmesi tavsiye edilir.



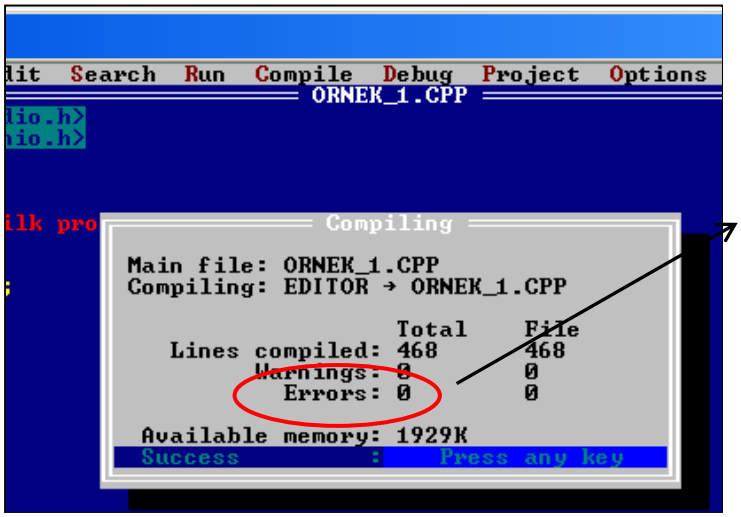
Ekran ile ilgili düzenlemeleri yapan önişlemci

Program sonuçlarının ekranda kalmasını sağlayan fonksiyon

Program ve fonksiyonları sona erdiren komut



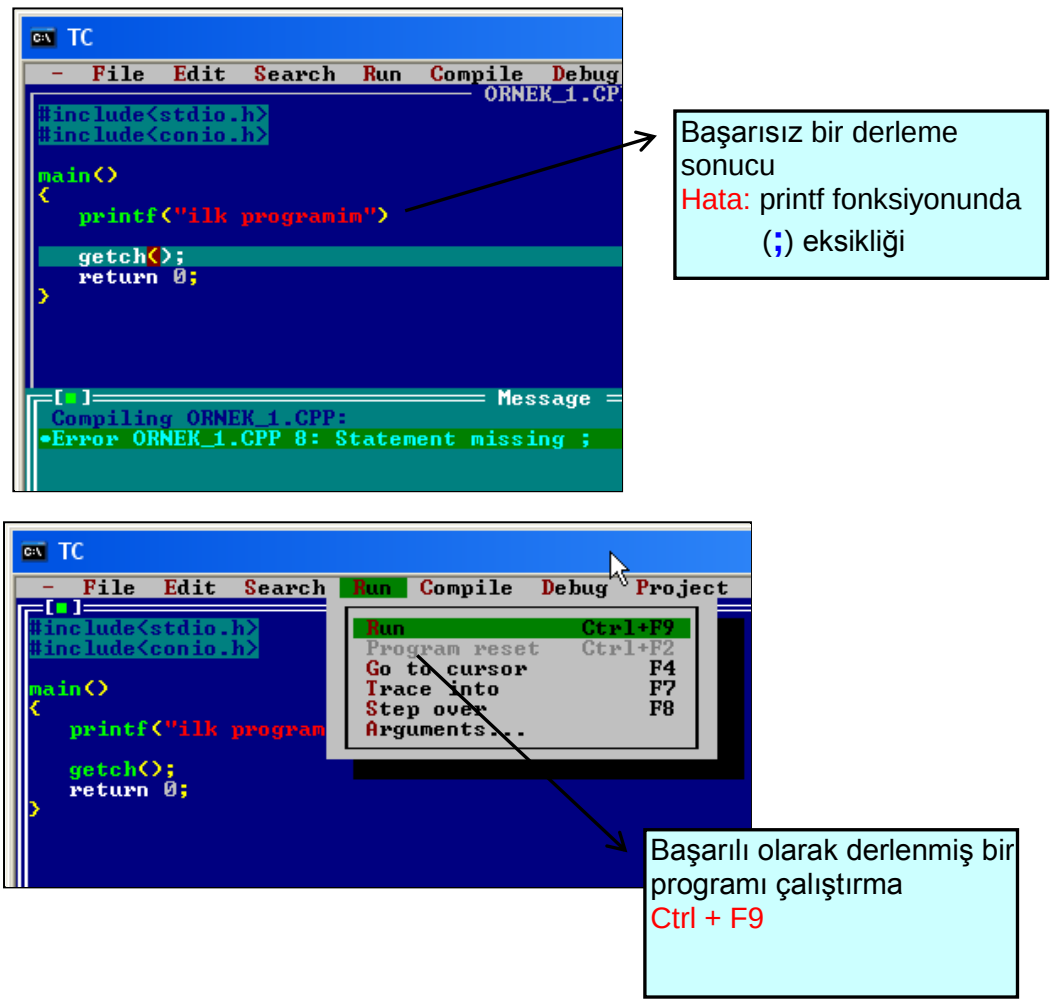
Derleme işlemi
Alt + F9

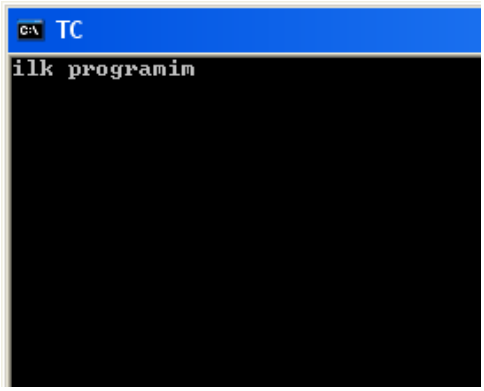


Başarılı bir derleme sonucu

Program komutları bilgisayar donanımlarına yönelik yazıldığından dolayı bunların hatasız olması gerekmektedir. Çoğu zaman hatalar, yanlış yazılmış bir komut, noktalama işaretlerinde yapılmış bir hata vb. olabilirler. Dolayısı ile hatalı yazılmış bir program kodu derlendiği zaman program çalışmaz ve yapılan hatalar editörün alt kısmında açılan bir pencerede belirtilir.

Genelde programlamada yapılan en yaygın hata ise spesifik yapı hatasıdır (syntax error) ve bu hata türü genellikle komut formatının yanlış yazılmasından kaynaklanır. Oluşan hataları gidermek için editörün altında bulunan hata penceresinde hataların oluştuğu satırlar tespit edilip daha sonra hataların düzeltilmesi yapılır. Eğer hata düzeltilemiyorsa C program dili ile ilgili bir referansa müracaat edilir. Başka bir yol ise programdaki hatanın olduğu yerler programdan çıkartılıp alternatif yazılım ile programın adım adım çalıştırılması ile bulunmasıdır. Bu işlem **F8** tuşu ile yapılır. Aşağıdaki şekilde editör üzerinde hata mesajlarının oluştuğu hata penceresi ile program kaynak kodu görülmektedir:





Uygulanabilir dosya
Ornek_1.exe
(temp dizini içindedir)

C dili için farklı derleyiciler kullanmak mümkündür. Günümüzde en popüler derleyiciler visual C++, visual C, gcc arasında sayılabilir.Bu derleyicilerden gcc ücretsiz olduğu için günümüzde çok kullanılan derleyiciler arasına girmiştir.
Aşağıda gcc ile ilgili ek bilgi verilmektedir.

GCC Kullanımı

"GCC" unix türevi işletim sistemlerinde kullanılan c derleyicisidir. "GNU C Complier" kelimelerinin baş harflerinden oluşmuştur. Gcc ücretsiz bir derleyicidir, bu nedenle kullanımı oldukça yaygındır.

Gcc'nin bir çok versiyonu vardır. Unix türevi işletim sitemi kullanılan bilgisayarınızdaki gcc versiyonunu görüntülemek istediğinizde aşağıdaki komutu kullanabilirsiz.

```
# gcc –v
```

bu komutu yazdığınızda karşınıza gcc programının nasıl konfigüre edildiği ve versiyon bilgileri karşınıza gelecektir.

Şimdi ilk gcc kullanımını öğrenmek amacı ile aşağıda verilen kodu yazıp bu kodu "merhaba.c" olarak kaydedelim.

Not : Bu programı herhangi bir unix editör programında yazabilirsiniz. Örneğin "vi", "pico", "gedit" gibi.

```
#include <stdio.h>
main()
{
        printf(" Merhaba Dünya");
}
```

Yukarıda yazdığımız "merhaba.c" dosyasını derleyelim. En basit şekli ile derlemek istenirse aşağıdaki komut kullanılabilir.

```
#gcc merhaba.c
```

Eğer herhangi bir hata mesajı ile karşılaşmadıysak derleme işlemi başarılı olmuş demektir. Bu komut satırı şeklinde derlenirse derleyici otomatik olarak çalıştırılabilir dosya türünde "a.out" adında bir dosya oluşturacaktır. Bu dosyayı Çalıştırmak içinde aşağıdaki komut yazılabilir.

```
# ./a.out
```

Bu durumda ekrana "Merhaba Dünya" yazdırılacaktır. Başka bir c dosyası oluşturup aynı şekilde derlendiğinde her derleme sonunda "a.out" dosyası oluşturulacaktır. Bu durumda sadece son derlenen dosyayı çalıştırabiliriz. Bu karışıklığı engellemek amacı ile "-o" parametresinini kullanabiliriz. "-o" parametresinin bitişiğine çalıştırılabilir türden oluşturulacak dosyanın ismi yazılır. Aşağıda gcc derleyicisinin "-o" parametresinin kullanımı ile ilgili örneği görebilrsiniz.

```
#gcc merhaba.c –o merhaba
```

Bu durumda gcc "merhaba.c" dosyasını derleyerek bu derleme sonunda "merhaba" adında bir dosya oluşturulacaktır. Bu dosyayı çalıştırmak için aşağıdaki formatta yazabilirsiniz.

```
#./mrehaba
```

Böylelikle oluşturulan dosyayı çalıştırmış oluruz.
Şimdide aşağıdaki programı derleyelim. Dosya adını "matematik.c" verelim.

```
#include<stdio.h>
#include<math.h>
main(){
        int a=80;
        printf("%d derecenin Kosinusu=",a,cos(a));
}
```

Derleme sırasında aşağıdaki düzeni kullanacağız.

```
# gcc -o  matematik matematik.c  -lm
```

Burada derleme komut dizisinde eklediğimiz –lm parametresi matematik kütüphanesi ile bağlantıyı sağlar.
Sadece math.h dosyasında değil diğer bazı kütüphaneler de eklendiğinde derleme komutuna da bazı eklentiler yapmak gerekir. Örneğin aşağıda yazdığımız "nc.c" dosyasında ncurses.h kütüphanesi kullanılmış.

```
#include<ncurses.h>
main(){
      initscr();
        printw("Merhaba Dünya ");
        refresh();
        getch();
        endwin();
}
```

"nc.c" programını derlemek için yazacağımız komutlar ve eklentileri aşağıdaki gibi olacaktır:

```
#gcc –o nc nc.c –I/usr/local/include –lncurses
```

Burada –lncurses -ncurses kütüphanesi ile link kurulacağını ve -I/usr/local/include satırı ile de kullanılan kütüphanelerin bulunduğu klasör yol olarak gösterilir.

Tabi sonuç olarak nc adlı çalıştırılabilir dosya oluşturulur. Çalıştırmak için aşağıdaki komut kullanılabilir:

```
#./nc
```

Gcc derleyicisin daha bir çok ek parametresi kullanılabilir. Bunlarla ilgili daha geniş bilgiye web sitelerinde rahatlıkla ulaşabiliriz.

UYGULAMA FAALİYETİ

Turbo C editorünü çalıştırarak dosya açma, kaydetme ve çalıştırma işlemlerini yapınız.

İşlem Basamakları	Öneriler
➢ Turbo C editörünü açınız.	➢ Turbo c editörünü açtıktan sonra options bölümünden directories seceneğinin doğru ayarlanmış olması gerekir.
➢ Yeni bir dosya açınız.	
➢ Her hangi bir örnek program yazınız.	➢ Programın yazım kurallarına dikkat ediniz.
➢ Bu dosyayı kaydediniz.	➢ Programa isim verirken (**. , / + -)** karakterlerini kullanmamaya özen gösteriniz.
➢ Programı derleyerek çalışma testi yapınız.	➢ Derleme sonrasında hata var ise komutların doğru yazılıp yazılmadığını kontrol ediniz.
➢ Dosyanın kaydedilip edilmediğini kontrol ediniz	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Performans Değerlendirme	Evet	Hayır
1. Programlama yazım aracını bilgisayarınıza kurdunuz mu ?		
2. Programlama yazım aracı kurulumunda gerekli ayarlamaları yaptınız mı?		
3. Yeni bir dosya oluşturdunuz mu?		
4. Dosyayı doğru olarak kaydettiniz mi?		
5. Programı yazıp tekrar kaydettiniz mi?		
6. Programı derlediniz mi?		
7. Program doğru çalışıyor mu?		
8. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
9. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Binary (ikili) sayı sisteminde 6 haneli kaç farklı sayı yazabiliriz.
A) 2^5
B) 2^6
C) 2^9
D) 2^7

2. Binary 1101 sayısının desimal (onluk) sayı sistemindeki karşılığı aşağıdakilerden hangisidir?
A) 13
B) 14
C) 15
D) 16

3. Aşağıdaki derleyicilerden hangisi unix türevi işletim sistemlerinde kullanılan c derleyicisidir?
A) Bcc
B) Tcp
C) Gcc
D) turbo c

4. Aşağıdaki programlama dili-tip eşleşmelerinden hangisi **yanlıştır**. ?
A) Visualbasic- Görsel dil
B) Java – görsel dil
C) Assembly- Düşük seviyeli programlama dili
D) C – görsel dil

5. Turbo C de program içerisinde dikkate alınamamasını istediğimiz satırların sonuna ve başına hangi işareti koymalıyız?

	Başlangıç	*Son*
A)	/-	-/
B)	#	#
C)	<?	?>
D)	/*	*/

6. Turbo C editöründe aşağıdaki menü ve alt seçeneklerin hangisini seçerek programı derleyebiliriz?
A) Compile- Compile seçeneği
B) File- New seçeneği
C) Options-directories seçeneği
D) Compile - Link seçeneği

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modülde yaptığınız uygulamaları tekrar yapınız.

Modülde kazandığınız becerileri aşağıdaki tablo doğrultusunda ölçünüz.

Performans Değerlendirme	Evet	Hayır
Problemi doğru tespit ettiniz mi?		
Değişken isimlerini belirlendiniz mi?		
Akış diyagramını çizdiniz mi?		
Komut listesini yazdınız mı?		
Akış diyagramını ve komut listesinin uyumunu kontrol ettiniz mi?		
Programlama yazım aracını bilgisayarınıza kurdunuz mu?		
Programlama yazım aracı kurulumunda gerekli ayarlamaları yaptınız mı?		
Yeni bir dosya oluşturdunuz mu?		
Dosyayı doğru olarak kaydettiniz mi?		
Programı yazıp tekrar kaydettiniz mi?		
Programı derlediniz mi?		
Program doğru çalışıyor mu?		
Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
Süreyi iyi kullandınız mı?		

Modül değerlendirmeniz sonucunda hayır seçeneğini işaretlediğiniz işlemleri tekrar ediniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ–1'İN CEVAP ANAHTARI

1	R1+R2		
2	C	D	E
	2	3	4
3	TABAN*YÜKSEKLİK /2		
4	C		

ÖĞRENME FAALİYETİ–2'İN CEVAP ANAHTARI

1	B
2	A
3	C
4	D
5	D
6	A

KAYNAKÇA

- ISHIDA Yasuhiro, Hideki MURAKAMI, Ito KOICHI, Gürcan ÇAYIR, **Bilgisayar Kontrol Teknolojisi**, M.E.B–JICA, 2005.